

# NI Vision

## NI Vision for Visual Basic User Manual

## Worldwide Technical Support and Product Information

[ni.com](http://ni.com)

### National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

### Worldwide Offices

Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599,  
Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00,  
Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000,  
Israel 972 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400,  
Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466,  
New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210,  
Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197,  
Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222,  
Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at [ni.com/info](http://ni.com/info) and enter the info code `feedback`.

# Important Information

---

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

## Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on [ni.com/legal](http://ni.com/legal) for more information about National Instruments trademarks.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

## Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or [ni.com/patents](http://ni.com/patents).

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Contents

---

## About This Manual

Conventions .....	ix
Related Documentation.....	x

## Chapter 1

### Introduction to NI Vision

About NI .....	1-1
NI Vision for Visual Basic Organization.....	1-1
cwimaq.ocx.....	1-1
CWIMAQ Control .....	1-2
CWIMAQVision Control.....	1-2
CWIMAQViewer Control.....	1-2
niocr.ocx .....	1-2
NIOCR Control .....	1-3
cwmv.ocx.....	1-3
CWMachineVision Control .....	1-3
ActiveX Objects .....	1-4
Creating NI Vision Applications .....	1-4

## Chapter 2

### Getting Measurement-Ready Images

Set Up Your Imaging System .....	2-1
Calibrate Your Imaging System .....	2-2
Create an Image .....	2-2
Acquire or Read an Image .....	2-4
Acquiring an Image .....	2-4
One-Shot Acquisition.....	2-4
Continuous Acquisition.....	2-5
Reading a File.....	2-6
Converting an Array to an Image .....	2-6
Display an Image .....	2-6
Attach Calibration Information.....	2-7
Analyze an Image .....	2-7
Improve an Image .....	2-9
Lookup Tables .....	2-9
Filters.....	2-9
Convolution Filter.....	2-10
Nth Order Filter.....	2-10

Grayscale Morphology .....	2-10
FFT .....	2-11
Complex Image Operations .....	2-12

## Chapter 3

### Making Grayscale and Color Measurements

Define Regions of Interest .....	3-1
Defining Regions Interactively .....	3-1
Defining Regions Programmatically .....	3-5
Defining Regions with Masks .....	3-6
Measure Grayscale Statistics .....	3-6
Measure Color Statistics .....	3-7
Comparing Colors .....	3-9
Learning Color Information .....	3-9
Specifying the Color Information to Learn .....	3-10
Using the Entire Image .....	3-10
Choosing a Color Representation Sensitivity .....	3-12
Ignoring Learned Colors .....	3-13

## Chapter 4

### Performing Particle Analysis

Create a Binary Image .....	4-1
Improve the Binary Image .....	4-2
Removing Unwanted Particles .....	4-3
Separating Touching Particles .....	4-4
Using Watershed Transform .....	4-4
Using Binary Morphology .....	4-4
Improving Particle Shapes .....	4-4
Make Particle Measurements .....	4-5

## Chapter 5

### Performing Machine Vision Tasks

Locate Objects to Inspect .....	5-2
Using Edge Detection to Build a Coordinate Transformation .....	5-3
Using Pattern Matching to Build a Coordinate Transformation .....	5-5
Choosing a Method to Build the Coordinate Transformation .....	5-7
Set Search Areas .....	5-8
Defining Regions Interactively .....	5-8
Defining Regions Programmatically .....	5-9

Find Measurement Points .....	5-9
Finding Features Using Edge Detection.....	5-9
Finding Lines or Circles.....	5-9
Finding Edge Points Along One Search Contour .....	5-11
Finding Edge Points Along Multiple Search Contours.....	5-12
Finding Points Using Pattern Matching .....	5-12
Defining and Creating Effective Template Images.....	5-13
Training the Pattern Matching Algorithm.....	5-15
Defining a Search Area .....	5-16
Setting Matching Parameters and Tolerances .....	5-17
Testing the Search Algorithm on Test Images.....	5-18
Using a Ranking Method to Verify Results .....	5-19
Finding Points Using Geometric Matching.....	5-19
Defining and Creating Effective Template Images.....	5-20
Training the Geometric Matching Algorithm .....	5-21
Setting Matching Parameters and Tolerances .....	5-22
Testing the Search Algorithm on Test Images.....	5-23
Finding Points Using Color Pattern Matching .....	5-24
Defining and Creating Effective Color Template Images .....	5-25
Training the Color Pattern Matching Algorithm.....	5-26
Defining a Search Area .....	5-27
Setting Matching Parameters and Tolerances .....	5-28
Testing the Search Algorithm on Test Images.....	5-30
Finding Points Using Color Location.....	5-30
Convert Pixel Coordinates to Real-World Coordinates.....	5-31
Make Measurements .....	5-31
Distance Measurements.....	5-31
Analytic Geometry Measurements .....	5-32
Instrument Reader Measurements .....	5-33
Identify Parts Under Inspection .....	5-33
Classifying Samples .....	5-33
Reading Characters.....	5-34
Reading Barcodes .....	5-34
Read 1D Barcodes.....	5-34
Read Data Matrix Code.....	5-35
Read QR Code .....	5-35
Read PDF417 Code.....	5-35
Inspect Image for Defects .....	5-35
Compare to Golden Template .....	5-35
Verify Characters.....	5-36
Display Results .....	5-36

## Chapter 6

### Calibrating Images

Perspective and Nonlinear Distortion Calibration.....	6-1
Defining a Calibration Template.....	6-2
Defining a Reference Coordinate System.....	6-3
Learning Calibration Information .....	6-5
Specifying Scaling Factors .....	6-6
Choosing a Region of Interest .....	6-6
Choosing a Learning Algorithm .....	6-6
Using the Learning Score .....	6-7
Learning the Error Map .....	6-8
Learning the Correction Table.....	6-8
Setting the Scaling Mode.....	6-8
Calibration Invalidation.....	6-9
Simple Calibration.....	6-9
Save Calibration Information .....	6-10
Attach Calibration Information .....	6-10

## Appendix A

### Technical Support and Professional Services

### Glossary

### Index

# About This Manual

---

This document is intended for engineers and scientists who have knowledge of Microsoft Visual Basic and need to create machine vision and image processing applications using Visual Basic objects. The manual guides you through tasks beginning with setting up the imaging system to taking measurements.

## Conventions

---

The following conventions appear in this manual:

»

The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.



This icon denotes a tip, which alerts you to advisory information.



This icon denotes a note, which alerts you to important information.

**bold**

Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

*italic*

Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

`monospace`

Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.



## Related Documentation

---

This manual assumes that you are familiar with Visual Basic and can use ActiveX controls in Visual Basic. The following are good sources of information about Visual Basic and ActiveX controls:

- [msdn.microsoft.com](http://msdn.microsoft.com)
- Documentation that accompanies Microsoft Visual Studio

In addition to this manual, the following documentation resources are available to help you create your vision application.

### NI Vision

- *NI Vision Concepts Manual*—Describes the basic concepts of image analysis, image processing, and machine vision. This document also contains in-depth discussions about imaging functions for advanced users.
- *NI Vision for Visual Basic Reference Help*—Contains reference information about NI Vision for Visual Basic objects and their associated properties, methods, and events.
- *NI OCR Training Interface Help*—Contains information about how to use the NI OCR Training Interface to train characters, save character sets, and verify characters by comparing them to a reference character.
- Example programs—Illustrate common applications you can create with Vision for Visual Basic. You can find these examples in the `Vision\Examples\MSVB` directory. You can find examples of how to create specific applications in Microsoft Visual Basic .NET in the `Vision\Examples\MSVB.NET` directory.

### NI Vision Assistant

- *NI Vision Assistant Tutorial*—Describes the NI Vision Assistant software interface and guides you through creating example image processing and machine vision applications.
- *NI Vision Assistant Help*—Contains descriptions of the NI Vision Assistant features and functions and provides instructions for using them.

## Other Documentation

- National Instruments image acquisition device user manuals—Contain installation instructions and device-specific information.
- *NI-IMAQ Help*—Contains fundamental programming concepts for NI-IMAQ driver software and terminology for using NI image acquisition devices.
- *NI-IMAQdx Help*—Contains fundamental programming concepts for NI-IMAQdx driver software and terminology for using IEEE 1394 and Gigabit Ethernet (GigE) cameras.
- NI Developer Zone (NIDZ)—Contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, a product advisor, and a community area where you can share ideas, questions, and source code with developers around the world. The NI Developer Zone is located on the National Instruments Web site at [ni.com/zone](http://ni.com/zone).

---

# Introduction to NI Vision

This chapter describes the NI Vision for Visual Basic software, outlines the NI Vision for Visual Basic architecture, and lists the steps for creating a machine vision application.



**Note** Refer to the *NI Vision Development Module Readme* for information about the system requirements and installation procedures for NI Vision for Visual Basic.

## About NI

---

NI Vision for Visual Basic is a collection of ActiveX controls you can use to develop machine vision and scientific imaging applications. The NI Vision Development Module also includes the same imaging functions for LabWindows™/CVI™ and other C development environments, as well as VIs for LabVIEW. Vision Assistant, another NI Vision Development Module software product, enables you to prototype your application strategy quickly without having to do any programming. Additionally, NI offers Vision Builder for Automated Inspection: configurable machine vision software that you can use to prototype, benchmark, and deploy applications.

## NI Vision for Visual Basic Organization

---

NI Vision for Visual Basic consists of five ActiveX controls contained in three files: `cwimaq.ocx`, `cwmv.ocx`, and `niocr.ocx`.

### **cwimaq.ocx**

`cwimaq.ocx` contains the following three ActiveX controls and a collection of ActiveX objects: `CWIMAQ`, `CWIMAQVision`, and `CWIMAQViewer`. Refer to the [ActiveX Objects](#) section for information about the ActiveX objects.

## CWIMAQ Control

Use this control to configure and perform an acquisition from the image acquisition device. The CWIMAQ control has property pages that allow you to modify various parameters to configure the acquisition and gather information about the image acquisition device. Most of the functionality available from the property pages during design time is also available through the properties of the CWIMAQ control during run-time. The control has methods that allow you to perform and control acquisitions, as well.



**Note** You must have the NI-IMAQ driver software installed on the target system to use the CWIMAQ control. For information about NI-IMAQ, refer to the *NI-IMAQ Help* that came with the NI image acquisition device.

## CWIMAQVision Control

Use this control to analyze and process images and their related data. The CWIMAQVision control provides methods for reading and writing images to and from files, analyzing images, and performing a variety of image processing algorithms on images.

## CWIMAQViewer Control

Use this control to display images and provide the interface through which the user will interact with the displayed image. This includes the ability to zoom and pan images and to draw regions of interest (ROIs) on an image. The CWIMAQViewer control has property pages that allow you to configure the viewer's appearance and behavior during design time as well as properties that you can configure during run-time. The control has methods that allow you to attach images to and detach images from the viewer for display purposes.



**Note** The CWIMAQViewer control is referred to as a *viewer* in the remainder of this document.

## niocr.ocx

`niocr.ocx` provides one ActiveX control and a collection of ActiveX objects you use in a machine vision application to perform optical character recognition (OCR).

## NIOCR Control

Use this control to perform OCR, which is the process by which the machine vision software reads text and/or characters in an image. OCR consists of the following two procedures:

- Training characters
- Reading characters

Training characters is the process by which you teach the machine vision software the types of characters and/or patterns you want to read in the image during the reading procedure. You can use NI Vision to train any number of characters, creating a character set, which is the set of characters that you later compare with objects during the reading procedure. You store the character set you create in a character set file. Training might be a one-time process, or it might be a process you repeat several times, creating several character sets to broaden the scope of characters you want to detect in an image.

Reading characters is the process by which the machine vision application you create analyzes an image to determine if the objects match the characters you trained. The machine vision application reads characters in an image using the character set that you created when you trained characters.

## CWMV.OCX

`cwmv.ocx` contains one ActiveX control and a collection of ActiveX objects. Refer to the [ActiveX Objects](#) section for more information about ActiveX objects.

## CWMachineVision Control

Use this control to perform high-level machine vision tasks, such as measuring distances. This control is written entirely in Visual Basic using the methods on the CWIMAQVision and CWIMAQViewer controls. The source code for the CWMachineVision control is included in the product. For more information about CWMachineVision methods, refer to Chapter 5, [Performing Machine Vision Tasks](#).



**Tip** Refer to the source code of the CWMachineVision control for an example of how to use the CWIMAQVision methods.

## ActiveX Objects

Use the objects to group related input parameters and output parameters to certain methods, thus reducing the number of parameters that you actually need to pass to those methods.



**Note** ActiveX objects in `cwimaq.ocx` have a CWIMAQ prefix, objects in `nioocr.ocx` have an NIOCR prefix, and objects in `cwmv.ocx` have a CWMV prefix.

You must create an ActiveX object before you can use it. You can use the `New` keyword in Visual Basic to create these objects. For example, use the following syntax to create and store an image in a variable named `image`:

```
Dim image As New CWIMAQImage
```



**Tip** If you intend to develop an application in Visual C++, National Instruments recommends that you use NI Vision for LabWindows/CVI. However, if you decide to use NI Vision for Visual Basic to develop applications for Visual C++, you can create objects using the respective `Create` methods on the `CWIMAQVision` control or `CWMachineVision` control. For example, to create a `CWIMAQImage` object, use the `CWIMAQVision.CreateCWIMAQImage` method.

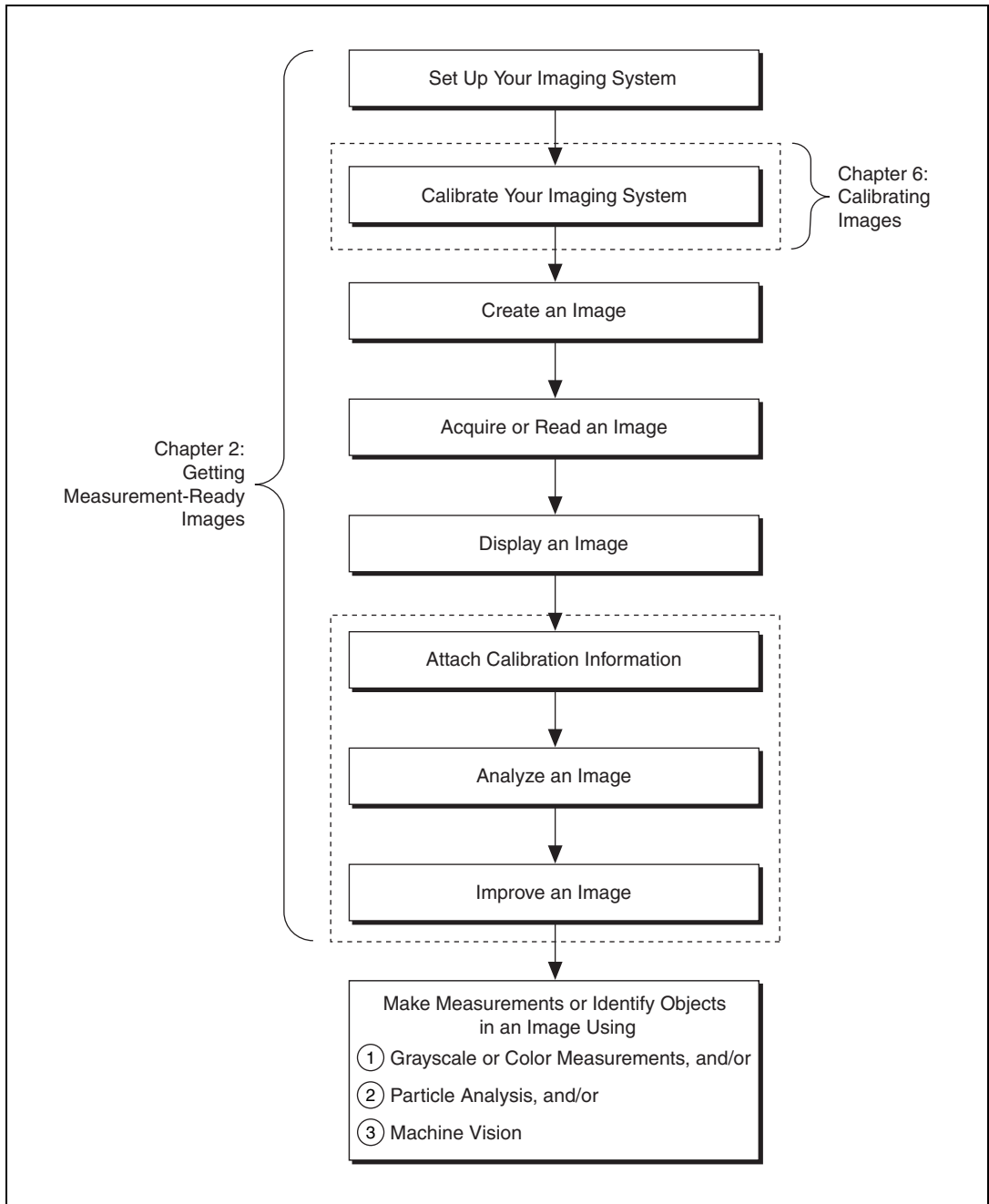
## Creating NI Vision Applications

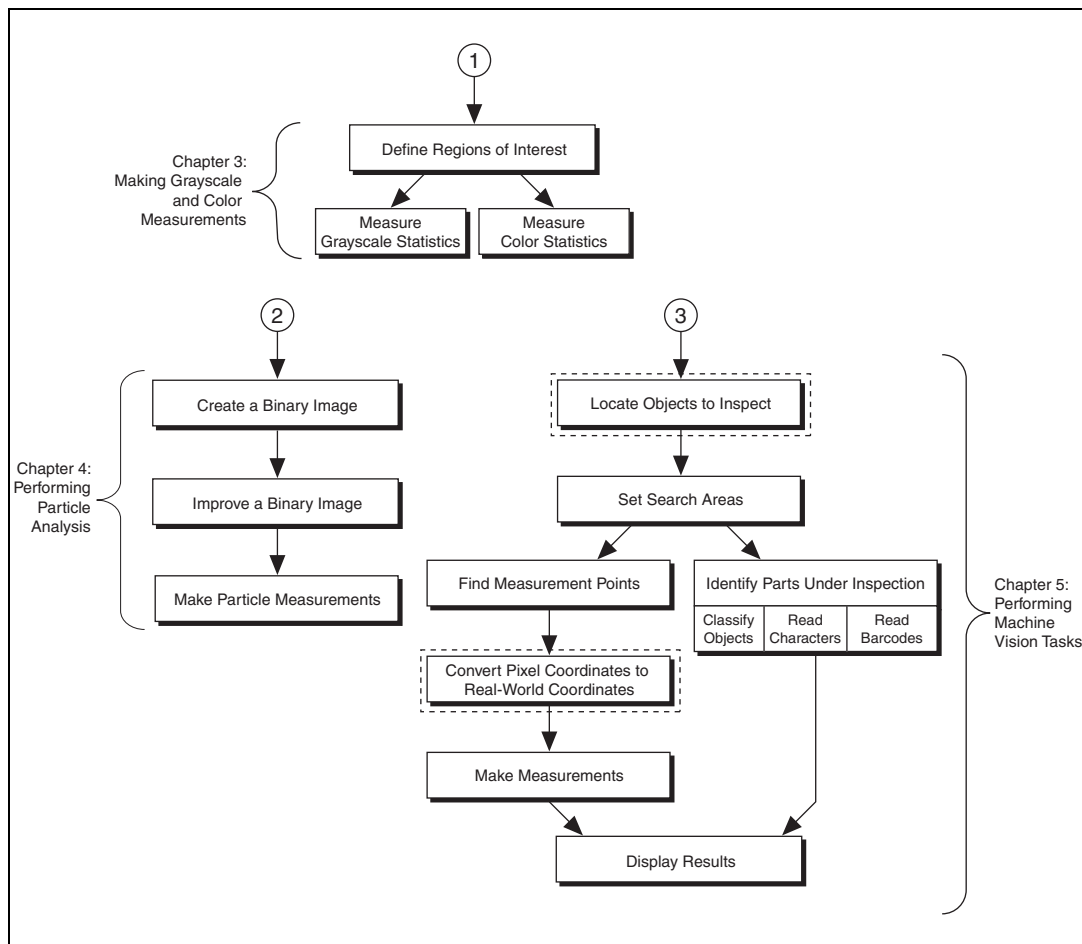
---

Figures 1-1 and 1-2 illustrate the steps for creating an application with NI Vision. Figure 1-1 describes the general steps for designing an NI Vision application. The last step in Figure 1-1 is expanded upon in Figure 1-2. You can use a combination of the items in the last step to create a NI Vision application. For more information about items in either diagram, refer to the corresponding chapter listed beside the item.



**Note** Diagram items enclosed with dashed lines are optional steps.

**Figure 1-1.** General Steps for Designing a Vision Application



**Figure 1-2.** Inspection Steps for Building a Vision Application



---

# Getting Measurement-Ready Images

This chapter describes how to set up an imaging system, acquire and display an image, analyze the image, and prepare the image for additional processing.

## Set Up Your Imaging System

---

Before you acquire, analyze, and process images, you must set up an imaging system. The manner in which you set up the system depends on the imaging environment and the type of analysis and processing you need to do. Your imaging system should produce images with high enough quality so that you can extract the information you need from the images.

Follow the guidelines below to set up an imaging system.

1. Determine the type of equipment you need based on the space constraints and the size of the object you need to inspect. For more information, refer to Chapter 3, *System Setup and Calibration*, of the *NI Vision Concepts Manual*.
  - a. Make sure the camera sensor is large enough to satisfy the minimum resolution requirement.
  - b. Make sure the lens has a depth of field high enough to keep all of the objects in focus regardless of their distance from the lens. Also, make sure the lens has a focal length that meets your needs.
  - c. Make sure the lighting provides enough contrast between the object under inspection and the background for you to extract the information you need from the image.
2. Position the camera so that it is parallel to the object under inspection. If the camera acquires images of the object from an angle, perspective errors occur. Even though you can compensate for these errors with software, NI recommends that you use a perpendicular inspection angle to obtain the fastest and most accurate results.
3. Select an image acquisition device that meets your needs. National Instruments offers several image acquisition devices, such as analog

color and monochrome devices as well as digital devices. Visit [ni.com/vision](http://ni.com/vision) for more information about NI image acquisition devices.



4. Configure the driver software for the image acquisition device. If you have a National Instruments image acquisition device, configure the NI-IMAQ or NI-IMAQdx driver software through Measurement & Automation Explorer (MAX). Open MAX by double-clicking the Measurement & Automation Explorer icon on the desktop. For more information, refer to the *Measurement & Automation Explorer for NI-IMAQ Help* and *Measurement & Automation Explorer for NI-IMAQdx Help*.

## Calibrate Your Imaging System

---

After you set up the imaging system, you may want to calibrate the system. Calibrate the imaging system to assign real-world coordinates to pixel coordinates and compensate for perspective and nonlinear errors inherent in the imaging system.

Perspective errors occur when the camera axis is not perpendicular to the object under inspection. Nonlinear distortion may occur from aberrations in the camera lens. Perspective errors and lens aberrations cause images to appear distorted. This distortion displaces information in an image, but it does not necessarily destroy the information in the image.

Use simple calibration if you want only to assign real-world coordinates to pixel coordinates. Use perspective and nonlinear distortion calibration if you need to compensate for perspective errors and nonlinear lens distortion. For detailed information about calibration, refer to Chapter 6, [Calibrating Images](#).

## Create an Image

---

The `CWIMAQImage` object encapsulates all the information required to represent an image.



**Note** `CWIMAQImage` is referred to as an *image* in the remainder of this document.

An image can be one of many types, depending on the data it stores. The following image types are valid:

- 8-bit
- 16-bit

- Single-precision floating point
- Complex
- 32-bit RGB
- 32-bit HSL
- 64-bit RGB

When you create an image, it is an 8-bit image by default. You can set the `Type` property on the image object to change the image type.

When you create an image, no memory is allocated to store the image pixels. NI Vision methods automatically allocate the appropriate amount of memory when the image size is modified. For example, methods that acquire or resample an image alter the image size, so they allocate the appropriate memory space for the image pixels.

Most methods belonging to the NI Vision library require an input of one or more image objects. The number of images a method takes depends on the image processing function and the type of image you want to use.

NI Vision methods that analyze the image but do not modify the image contents require the input of one source image. Methods that process the contents of the image require one or more source images and a destination image. Exceptions to the preceding statements are methods that take a mask image as input.

The presence of a `MaskImage` parameter indicates that the processing or analysis is dependent on the contents of the mask image. The only pixels in the source image that are processed are those whose corresponding pixels in the mask image are non-zero. If a mask image pixel is 0, the corresponding source image pixel is not processed or analyzed.



**Note** The image mask must be an 8-bit image for all NI Vision methods except `CWIMAQVision.Quantify`, which supports both 8-bit and 16-bit image masks.

If you want to apply a processing or analysis method to the entire image, do not supply the optional mask image. Using the same image for both the source image and mask image also has the same effect as not using the mask image, except in this case the source image must be an 8-bit image.

Most operations between two images require that the images have the same type and size. However, arithmetic operations work between two different types of images. For example, an arithmetic operation between an 8-bit image and 16-bit image results in a 16-bit image.

## Acquire or Read an Image

---

After you create an image, you can acquire an image into the imaging system in one of the following three ways:

- Acquire an image with a camera through the image acquisition device.
- Load an image from a file stored on the computer.
- Convert the data stored in a 2D array to an image.

Methods that acquire images, load images from file, or convert data from a 2D array automatically allocate the memory space required to accommodate the image data.

### Acquiring an Image

Use the CWIMAQ control to acquire images with a National Instruments image acquisition device. You can use NI Vision for Visual Basic to perform one-shot and continuous acquisitions. You can choose the acquisition type during design time by setting the value of the **Acquisition Type** combo box to **One-Shot** or **Continuous**. The **Acquisition Type** combo box is located on the Acquisition property page of the CWIMAQ control. You can set the value at run-time by setting the `CWIMAQ.AcquisitionType` property to `cwimaqAcquisitionOneShot` or `cwimaqAcquisitionContinuous`.

### One-Shot Acquisition

Use a one-shot acquisition to start an acquisition, perform the acquisition, and stop the acquisition using a single method. The number of frames acquired is equal to the number of images in the images collection. Use the `CWIMAQ.AcquireImage` method to perform this operation synchronously. Use the `CWIMAQ.Start` method to perform this operation asynchronously. For information about synchronous and asynchronous acquisitions, refer to the *NI-IMAQ Help*.

If you want to acquire a single field or frame into a buffer, set the image count to 1. This operation is also referred to as a *snap*. Use a snap for low-speed or single capture applications. The following code illustrates a synchronous snap:

```
Private Sub Start_Click()  
    CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot  
    CWIMAQ1.AcquireImage  
End Sub
```

If you want to acquire multiple frames, set the image count to the number of frames you want to acquire. This operation is called a *sequence*. Use a sequence for applications that process multiple images. The following code illustrates an asynchronous sequence, where `numberOfImages` is the number of images that you want to process:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot
    CWIMAQ1.Images.RemoveAll
    CWIMAQ1.Images.Add numberOfImages
    CWIMAQ1.Start
End Sub
```

## Continuous Acquisition

Use a continuous acquisition to start an acquisition and continuously acquire frames into the image buffers, and then explicitly stop the acquisition. Use the `CWIMAQ1.Start` method to start the acquisition. Use the `CWIMAQ1.Stop` method to stop the acquisition. If you use a single buffer for the acquisition, this operation is called a *grab*. The following code illustrates a grab:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType =_
    cwimaqAcquisitionContinuous
    CWIMAQ1.Start
End Sub
```

```
Private Sub Stop_Click()
    CWIMAQ1.Stop
End Sub
```

A *ring* operation uses multiple buffers for the acquisition. Use a ring for high-speed applications that require processing on every image. The following code illustrates a ring, where `numberOfImages` is the number of images that you want to process:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType =_
    cwimaqAcquisitionContinuous
    CWIMAQ1.Images.RemoveAll
    CWIMAQ1.Images.Add numberOfImages
    CWIMAQ1.Start
End Sub
```

```
Private Sub Stop_Click()  
    CWIMAQ1.Stop  
End Sub
```

## Reading a File

Use the `CWIMAQVision.ReadImage` method to open and read data from a file stored on the computer into the image reference. You can read from image files stored in several standard formats, such as BMP, TIFF, JPEG, JPEG2000, PNG, and AIPD. In all cases, the software automatically converts the pixels it reads into the type of image you pass in.

Use the `CWIMAQVision.ReadImageAndVisionInfo` method to open an image file containing additional information, such as calibration information, template information for pattern matching, or overlay information. For more information about pattern matching templates and overlays, refer to Chapter 5, *Performing Machine Vision Tasks*.

You also can use the `CWIMAQVision.GetFileInformation` method to retrieve image properties—image size, pixel depth, recommended image type, and calibration units—without actually reading all the image data.

## Converting an Array to an Image

Use the `CWIMAQImage.ArrayToImage` method to convert an array to an image. You also can use the `CWIMAQImage.ImageToArray` method to convert an image to an array.

## Display an Image

---

Display an image using the `CWIMAQViewer` control. Use `CWIMAQViewer.Attach` to attach the image you want the viewer to display. When you attach an image to a viewer, the image automatically updates the viewer whenever an operation modifies the contents of the image. You can access the image attached to the viewer using the `CWIMAQViewer.Image` property. Before you attach an image to the viewer, the viewer already has an image attached by default. Therefore, the viewer has an image attached to it at all times. You can use the attached image as either a source image, destination image, or both using the `CWIMAQViewer.Image` property.

You can use the `CWIMAQViewer.Palette` property to access the `CWIMAQPalette` object associated with the viewer. Use the `CWIMAQPalette` object to programmatically apply a color palette to

the viewer. You can set the `CWIMAQPalette.Type` property to apply predefined color palettes. For example, if you need to display a binary image—an image that contains particle regions with pixel values of 1 and a background region with pixel values of 0—set the `Type` property to `cwimaqPaletteBinary`. For more information about color palettes, refer to Chapter 2, *Display*, of the *NI Vision Concepts Manual*.

You also can set a default palette during design time using the Menu property page. Users can change the color palette during run time by using the right-click menu on the viewer.

## Attach Calibration Information

---

If you want to attach the calibration information of the current setup to each image you acquire, use

`CWIMAQVision.SetCalibrationInformation2`. This method takes in a source image that contains the calibration information and a destination image that you want to calibrate. The output image is the destination image with the calibration information attached to it. For detailed information about calibration, refer to Chapter 6, *Calibrating Images*.



**Note** Because calibration information is part of the image, it is propagated throughout the processing and analysis of the image. Methods that modify the image size, such as geometrical transforms, void the calibration information. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image and all of the attached calibration information to a file.

## Analyze an Image

---

When you acquire and display an image, you may want to analyze the contents of the image for the following reasons:

- To determine if the image quality is high enough for the inspection task.
- To obtain the values of parameters that you want to use in processing methods during the inspection process.

The histogram and line profile tools can help you analyze the quality of the images.

Use `CWIMAQVision.Histogram2` to analyze the overall grayscale distribution in the image. Use the histogram of the image to analyze two important criteria that define the quality of an image—saturation and contrast. If the image does not have enough light, the majority of the pixels will have low intensity values, which appear as a concentration of peaks on the left side of the histogram. If the image has too much light, the majority of the pixels will have a high intensity values, which appear as a concentration of peaks on the right side of the histogram. If the image has an appropriate amount of contrast, the histogram will have distinct regions of pixel concentrations. Use the histogram information to decide if the image quality is high enough to separate objects of interest from the background.

If the image quality meets your needs, use the histogram to determine the range of pixel values that correspond to objects in the image. You can use this range in processing methods, such as determining a threshold range during particle analysis.

If the image quality does not meet your needs, try to improve the imaging conditions to get the appropriate image quality. You may need to re-evaluate and modify each component of the imaging setup: lighting equipment and setup, lens tuning, camera operation mode, and acquisition board parameters. If you reach the best possible conditions with the setup but the image quality still does not meet your needs, try to improve the image quality using the image processing techniques described in the [Improve an Image](#) section of this chapter.

Use `CWIMAQVision.LineProfile2` to get the pixel distribution along a line in the image, or use `CWIMAQVision.RegionsProfile` to get the pixel distribution along a one-dimensional path in the image. By looking at the pixel distribution, you can determine if the image quality is high enough to provide you with sharp edges at object boundaries. Also, you can determine if the image is noisy, and identify the characteristics of the noise.

If the image quality meets your needs, use the pixel distribution information to determine some parameters of the inspection methods you want to use. For example, use the information from the line profile to determine the strength of the edge at the boundary of an object. You can input this information into `CWIMAQVision.FindEdges2` to find the edges of objects along the line.



# Improve an Image

---

Using the information you gathered from analyzing the image, you may want to improve the quality of the image for inspection. You can improve the image with lookup tables, filters, grayscale morphology, and Fast Fourier transforms (FFT).

## Lookup Tables

Apply *lookup table* (LUT) transformations to highlight image details in areas containing significant information at the expense of other areas. A LUT transformation converts input grayscale values in the source image into other grayscale values in the transformed image. NI Vision provides four methods that directly or indirectly apply lookup tables to images:

- `CWIMAQVision.MathLookup`—Converts the pixel values of an image by replacing them with values from a predefined lookup table. NI Vision has seven predefined lookup tables based on mathematical transformations. For more information about these lookup tables, refer to Chapter 5, *Image Processing*, in the *NI Vision Concepts Manual*.
- `CWIMAQVision.UserLookup`—Converts the pixel values of an image by replacing them with values from a user-defined lookup table.
- `CWIMAQVision.Equalize2`—Distributes the grayscale values evenly within a given grayscale range. Use this method to increase the contrast in images containing few grayscale values.
- `CWIMAQVision.Inverse`—Inverts the pixel intensities of an image to compute the negative of the image. For example, use this method before applying an automatic threshold to the image if the background pixels are brighter than the object pixels.

## Filters

Filter the image when you need to improve the sharpness of transitions in the image or increase the overall signal-to-noise ratio of the image. You can choose either a lowpass or highpass filter, depending on your needs.

Lowpass filters remove insignificant details by smoothing the image, removing sharp details, and smoothing the edges between the objects and the background. You can use `CWIMAQVision.LowPass` or define your own lowpass filter with `CWIMAQVision.Convolute` or `CWIMAQVision.NthOrder`.

Highpass filters emphasize details, such as edges, object boundaries, or cracks. These details represent sharp transitions in intensity value.

You can define your own highpass filter with `CWIMAQVision.Convolute` or `CWIMAQVision.NthOrder`, or you can use a predefined highpass filter with `CWIMAQVision.EdgeFilter` or `CWIMAQVision.CannyEdgeFilter`. `CWIMAQVision.EdgeFilter` allows you to find edges in an image using predefined edge detection kernels, such as the Sobel, Prewitt, and Roberts kernels.

## Convolution Filter

`CWIMAQVision.Convolute` allows you to use a predefined set of lowpass and highpass filters. Each filter is defined by a kernel of coefficients. Use the `CWIMAQKernel` object to define the filter. Use `CWIMAQKernel.LoadKernel` to load a predefined kernel into the object. If the predefined kernels do not meet your needs, use the `CWIMAQKernel.SetSize` method to set the size of the kernel and the `CWIMAQKernel.Element` property to set the data in the kernel.

## Nth Order Filter

`CWIMAQVision.NthOrder` allows you to define a lowpass or highpass filter depending on the value of  $N$  that you choose. One specific Nth order filter, the median filter, removes speckle noise, which appears as small black and white dots. For more information about Nth order filters, refer to Chapter 5, *Image Processing*, of the *NI Vision Concepts Manual*.

## Grayscale Morphology

Perform grayscale morphology when you want to filter grayscale features of an image. Grayscale morphology helps you remove or enhance isolated features, such as bright pixels on a dark background. Use these transformations on a grayscale image to enhance non-distinct features before thresholding the image in preparation for particle analysis.

Grayscale morphological transformations, which include erosions and dilations, compare a pixel to those pixels that surround it. An erosion keeps the smallest pixel values. A dilation keeps the largest pixel values.

For more information about grayscale morphology transformations, refer to Chapter 5, *Image Processing*, of the *NI Vision Concepts Manual*.

Use `CWIMAQVision.GrayMorphology` to perform one of the following seven transformations:

- Erosion—Reduces the brightness of pixels that are surrounded by neighbors with a lower intensity.
- Dilation—Increases the brightness of pixels surrounded by neighbors with a higher intensity. A dilation has the opposite effect of an erosion.
- Opening—Removes bright pixels isolated in dark regions and smooths boundaries.
- Closing—Removes dark pixels isolated in bright regions and smooths boundaries.
- Proper-opening—Removes bright pixels isolated in dark regions and smooths the inner contours of particles.
- Proper-closing—Removes dark pixels isolated in bright regions and smooths the inner contours of particles.
- Auto-median—Generates simpler particles that have fewer details.

## FFT

Use the Fast Fourier Transform (FFT) to convert an image into its frequency domain. In an image, details and sharp edges are associated with mid to high spatial frequencies because they introduce significant gray-level variations over short distances. Gradually varying patterns are associated with low spatial frequencies.

An image can have extraneous noise, such as periodic stripes, introduced during the digitization process. In the frequency domain, the periodic pattern is reduced to a limited set of high spatial frequencies. Also, the imaging setup may produce non-uniform lighting of the field of view, which produces an image with a light drift superimposed on the information you want to analyze. In the frequency domain, the light drift appears as a limited set of low frequencies around the average intensity of the image, which is the DC component.

You can use algorithms working in the frequency domain to isolate and remove these unwanted frequencies from the image. Complete the following steps to obtain an image in which the unwanted pattern has disappeared but the overall features remain:

1. Use `CWIMAQVision.FFT` to convert an image from the spatial domain to the frequency domain. This method computes the FFT of the image and results in a complex image representing the frequency information of the image.

2. Improve the image in the frequency domain with a lowpass or highpass frequency filter. Specify which type of filter to use with `CWIMAQVision.CxAttenuate` or `CWIMAQVision.CxTruncate`. Lowpass filters smooth noise, details, textures, and sharp edges in an image. Highpass filters emphasize details, textures, and sharp edges in images, but they also emphasize noise.
  - Lowpass attenuation—The amount of attenuation is directly proportional to the frequency information. At low frequencies, there is little attenuation. As the frequencies increase, the attenuation increases. This operation preserves all of the zero frequency information. Zero frequency information corresponds to the DC component of the image or the average intensity of the image in the spatial domain.
  - Highpass attenuation—The amount of attenuation is inversely proportional to the frequency information. At high frequencies, there is little attenuation. As the frequencies decrease, the attenuation increases. The zero frequency component is removed entirely.
  - Lowpass truncation—Specify a frequency. The frequency components above the ideal cutoff frequency are removed, and the frequencies below it remain unaltered.
  - Highpass truncation—Specify a frequency. The frequency components above the ideal cutoff frequency remain unaltered, and the frequencies below it are removed.
3. To transform the image back to the spatial domain, use `CWIMAQVision.InverseFFT`.

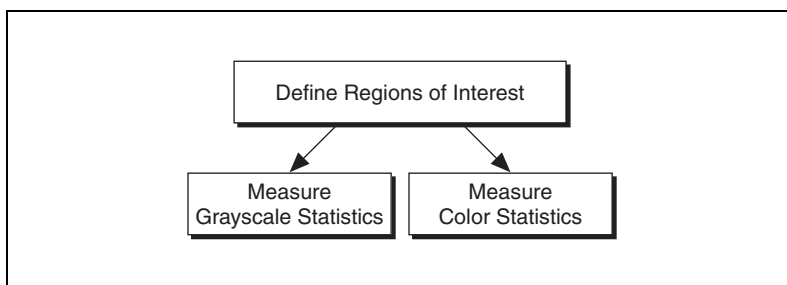
## Complex Image Operations

`CWIMAQVision.ReplaceComplexPlane` and `CWIMAQVision.ExtractComplexPlane` allow you to access, process, and update independently the magnitude, phase, real, and imaginary planes of a complex image. You can also convert a complex image to an array and back with `CWIMAQImage.ImageToArray` and `CWIMAQImage.ArrayToImage`.

---

# Making Grayscale and Color Measurements

This chapter describes how to take measurements from grayscale and color images. You can make inspection decisions based on image statistics, such as the mean intensity level in a region. Based on the image statistics, you can perform many machine vision inspection tasks on grayscale or color images, such as detecting the presence or absence of components, detecting flaws in parts, and comparing a color component with a reference. Figure 3-1 illustrates the basic steps involved in making grayscale and color measurements.



**Figure 3-1.** Steps to Taking Grayscale and Color Measurements

---











## Define Regions of Interest

An ROI is an area of an image in which you want to focus the image analysis. You can define an ROI interactively, programmatically, or with an image mask.




### Defining Regions Interactively

You can interactively define an ROI in a viewer that displays an image. Use the tools from the right-click menu to interactively define and manipulate the ROIs. Table 3-1 describes each of the tools and the manner in which you use them.

**Table 3-1.** Tools Palette Functions

Icon	Tool Name	Function
	Selection Tool	Select an ROI in the image and adjust the position of its control points and contours. Action: Click ROI or control points.
	Point	Select a pixel in the image. Action: Click the position of the pixel.
	Line	Draw a line in the image. Action: Click the initial position and click again at the final position.
	Rectangle	Draw a rectangle or square in the image. Action: Click one corner and drag to the opposite corner.
	Oval	Draw an oval or circle in the image. Action: Click the center position and drag to the required size.
	Polygon	Draw a polygon in the image. Action: Click to place a new vertex and double-click to complete the ROI element.
	Freehand Region	Draw a freehand region in the image. Action: Click the initial position, drag to the required shape and release the mouse button to complete the shape.
	Annulus	Draw an annulus in the image. Action: Click the center position and drag to the required size. Adjust the inner and outer radii, and adjust the start and end angle.
	Zoom	Zoom-in or zoom-out in an image. Action: Click the image to zoom in. Hold down the <Shift> key and click to zoom out.
	Pan	Pan around an image. Action: Click an initial position, drag to the required position and release the mouse button to complete the pan.

**Table 3-1.** Tools Palette Functions (Continued)

Icon	Tool Name	Function
	Broken Line	Draw a broken line in the image.  Action: Click to place a new vertex and double-click to complete the ROI element.
	Freehand Line	Draw a freehand line in the image.  Action: Click the initial position, drag to the required shape, and release the mouse button to complete the shape.
	Rotated Rectangle	Draw a rotated rectangle in the image.  Action: Click one corner and drag to the opposite corner to create the rectangle. Then click the lines inside the rectangle and drag to adjust the rotation angle.

Hold down <Shift> when drawing an ROI if you want to constrain the ROI to the horizontal, vertical, or diagonal axes, when possible. Use the selection tool to position an ROI by its control points or vertices. ROIs are context sensitive, meaning that the cursor actions differ depending on the ROI with which you interact. For example, if you move the cursor over the side of a rectangle, the cursor changes to indicate that you can click and drag the side to resize the rectangle. If you want to draw more than one ROI in a window, hold down <Ctrl> while drawing additional ROIs. You also can use `CWIMAQViewer.MaxContours` to set the maximum number of contours the viewer can have in its ROI.

In the status bar of the viewer, you can display tool information about the characteristics of ROIs you draw, as shown in Figure 3-2. Check the **Show Tool Info** check box on the Status Bar property page during design time, or set the `CWIMAQViewer.ShowToolInfo` property to `True` during run time to display tool information. You also can show or hide the tool information from the right-click menu.

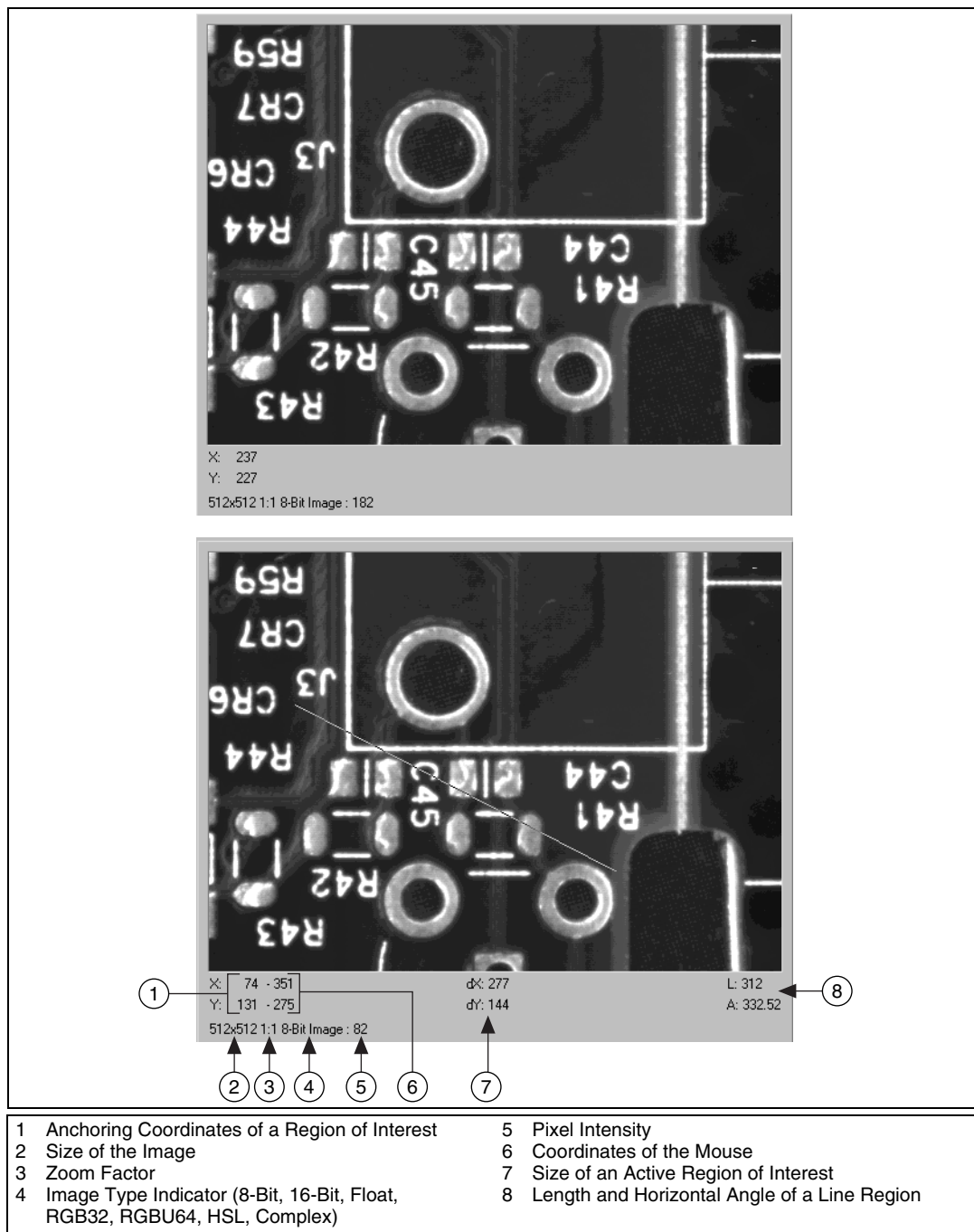


Figure 3-2. Tools Information



During design time, use the Menu property page to select which tools appear in the right-click menu. You also can designate a default tool from this property page. During run time, set the `CWIMAQViewer.MenuItems` to select the tools to display, and set `CWIMAQViewer.Tool` to select the default tool.

## Defining Regions Programmatically

You can define ROIs programmatically using the `CWIMAQRegions` collection. In NI Vision, shapes are represented by shape objects. For example, `CWIMAQPoint` represents a point, and `CWIMAQLine` represents a line. Use the following methods listed in Table 3-2 to add various shapes to the regions.

**Table 3-2.** Methods that Add Shapes to Regions

Method	Description
<code>AddPoint</code>	adds a point to the ROI
<code>AddLine</code>	adds a line to the ROI
<code>AddRectangle</code>	adds a rectangle to the ROI
<code>AddRotatedRectangle</code>	adds a rotated rectangle to the ROI
<code>AddOval</code>	adds an oval to the ROI
<code>AddAnnulus</code>	adds an annulus to the ROI
<code>AddBrokenLine</code>	adds a broken line to the ROI
<code>AddPolygon</code>	adds a polygon to the ROI
<code>AddFreeline</code>	adds a free line to the ROI
<code>AddFreeregion</code>	adds a free region to the ROI
<code>AddRegion</code>	adds a region object to the ROI

Use the `CWIMAQRegions.CopyTo` method to copy all the data from one `CWIMAQRegions` object to another.

You can define the regions on a viewer and access the regions using the `CWIMAQViewer.Regions` property.

The individual `CWIMAQRegion` objects provide access to the shapes in the collection. Each region has one shape object associated with it. Use the `CWIMAQRegion.Shape` property to determine what type of shape the

CWIMAQRegion contains. When you know the type of shape that the region contains, you can set the region into a shape variable and use that variable to manipulate the shape properties. For example, the following code resizes a rectangle selected on the viewer:

```
Dim MyRectangle As CWIMAQRectangle
Set MyRectangle = CWIMAQViewer1.Regions(1)
MyRectangle.Width = 100
MyRectangle.Height = 100
```

You also can pass CWIMAQRegion objects to any NI Vision method that takes a shape as a parameter. However, if the CWIMAQRegion does not contain the type of shape object that the method requires, a type mismatch error results.

## Defining Regions with Masks

You can define regions to process with image masks. An image mask is an 8-bit image of the same size as or smaller than the image you want to process. Pixels in the mask image determine if the corresponding pixel in the source image needs to be processed. If a pixel in the image mask has a value other than 0, the corresponding pixel in the source image is processed. If a pixel in the image mask has a value of 0, the corresponding pixel in the source image is left unchanged.

You can use a mask to define particles in a grayscale image when you need to make intensity measurements on those particles. First, threshold the image to make a new binary image. For more information about binary images, refer to Chapter 4, *Performing Particle Analysis*. You can input the binary image or a labeled version of the binary image as a mask image to the intensity measurement method. If you want to make color comparisons, convert the binary image into a CWIMAQRegions collection using `CWIMAQVision.MaskToRegions`.

## Measure Grayscale Statistics

---

You can measure grayscale statistics in images using light meters or quantitative analysis methods. You can obtain the center of energy for an image with the centroid method.

Use `CWMachineVision.LightMeterPoint` to measure the light intensity at a point in the image. Use `CWMachineVision.LightMeterLine` to get the pixel value statistics along a line in the image, such as mean intensity, standard deviation,

minimum intensity, and maximum intensity. Use `CWMachineVision.LightMeterRectangle` to get the pixel value statistics within a rectangular region in an image.

Use `CWIMAQVision.Quantify` to obtain the following statistics about the entire image or individual regions in the image: mean intensity, standard deviation, minimum intensity, maximum intensity, area, and the percentage of the image that you analyzed. You can specify regions in the image with a labeled image mask. A labeled image mask is a binary image that has been processed so that each region in the image mask has a unique intensity value. Use `CWIMAQVision.Label2` to label the image mask.

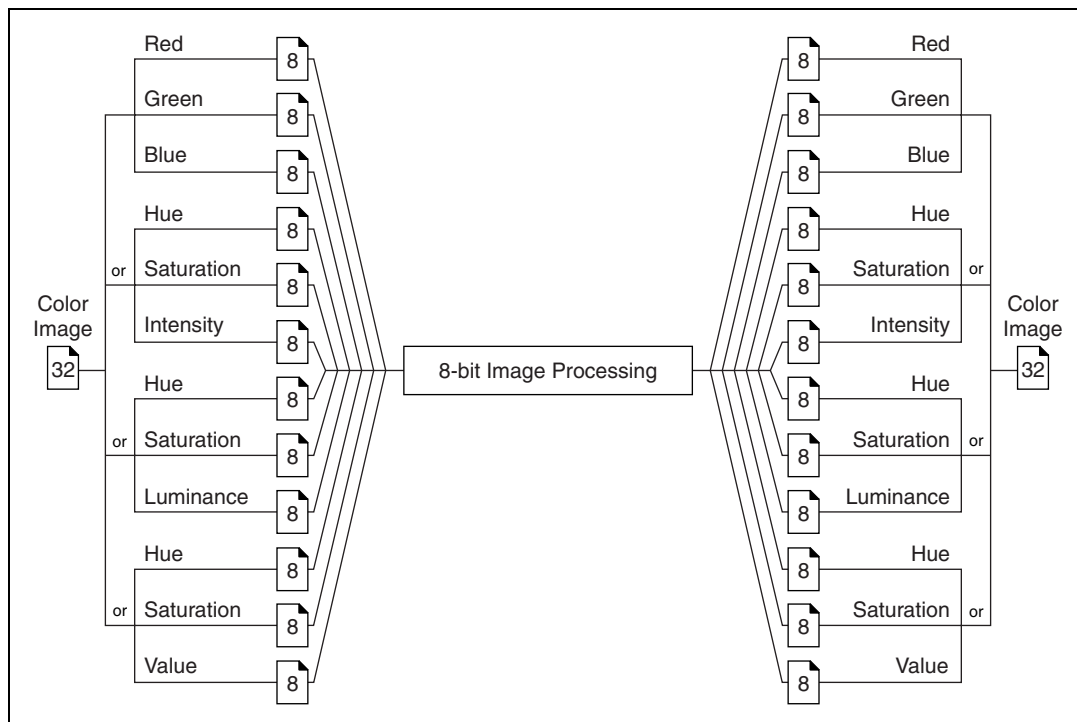
Use `CWIMAQVision.Centroid2` to compute the energy center of the image, or of a region within an image.

## Measure Color Statistics

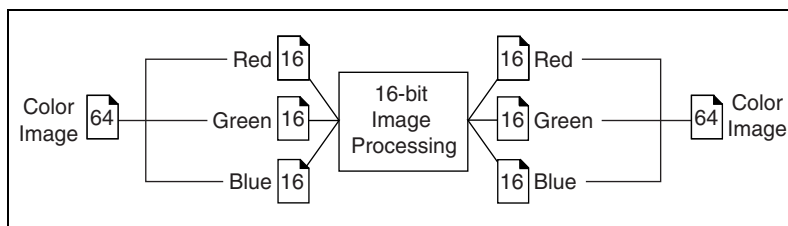
---

Most image processing and analysis methods apply to 8-bit and 16-bit images. However, you can analyze and process individual components of a color image.

Using `CWIMAQVision.ExtractColorPlanes`, you can break down a color image into various sets of primary components, such as RGB (Red, Green, and Blue), HSI (Hue, Saturation, and Intensity), HSL (Hue, Saturation, and Luminance), or HSV (Hue, Saturation, and Value). Each component becomes an 8-bit or 16-bit image that you can process like any other grayscale image. Use `CWIMAQVision.ExtractSingleColorPlane` to extract a single color plane from an image. Use `CWIMAQVision.ReplaceColorPlanes` to reassemble a color image from a set of three 8-bit or 16-bit images, where each image becomes one of the three primary components. Figures 3-3 and 3-4 illustrate how a color image breaks down into its three components.



**Figure 3-3.** Primary Components of a 32-Bit Color Image



**Figure 3-4.** Primary Components of a 64-Bit Color Image

A color pixel encoded as a Long value can be decomposed into its individual components using `CWIMAQVision.IntegerToColorValue`. You can convert a pixel value represented in any color model into its components in any other color model using `CWIMAQVision.ColorValueConversion2`.

## Comparing Colors

You can use the color matching capability of NI Vision to compare or evaluate the color content of an image or regions in an image.

Complete the following steps to compare colors using color matching:

1. Select an image containing the color information that you want to use as a reference. The color information can consist of a single color or multiple dissimilar colors, such as red and blue.
2. Use the entire image or regions in the image to learn the color information using `CWIMAQVision.LearnColor`, which stores the results of the operation in a `CWIMAQColorInformation` object that you supply as a parameter. The color information object has a color spectrum that contains a compact description of the color information that you learned. Refer to Chapter 15, *Color Inspection*, of the *NI Vision Concepts Manual* for more information. Use the `CWIMAQColorInformation` object to represent the learned color information for all subsequent matching operations.
3. Define an entire image, a region, or multiple regions in an image as the inspection or comparison area.
4. Use `CWIMAQVision.MatchColor` to compare the learned color information to the color information in the inspection regions. This method returns an array of scores that indicates how close the matches are to the learned color information.
5. Use the color matching score as a measure of similarity between the reference color information and the color information in the image regions being compared.

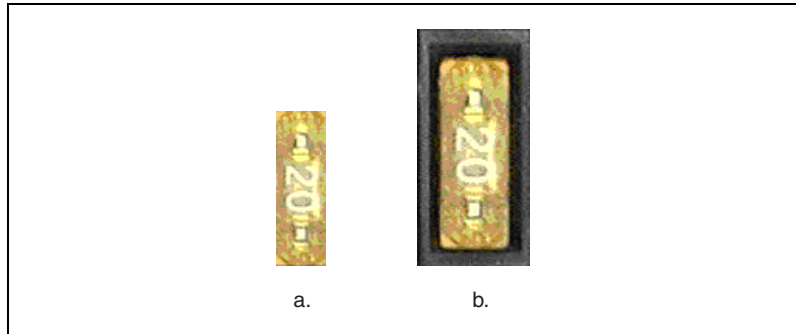
## Learning Color Information

When learning color information, choose the color information carefully:

- Specify an image or regions in an image that contain the color or color set that you want to learn.
- Specify the granularity required to represent the color information.
- Choose colors that you want to ignore during matching.

## Specifying the Color Information to Learn

Because color matching only uses color information to measure similarity, the image or regions in the image representing the object should contain only the significant colors that represent the object, as shown in Figure 3-5a. Figure 3-5b illustrates an unacceptable region containing background colors.

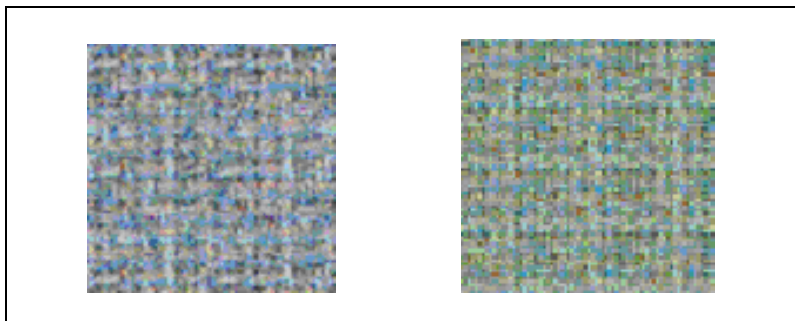


**Figure 3-5.** Template Color Information

The following sections specify when to learn the color information associated with an entire image, a region in an image, or multiple regions in an image.

## Using the Entire Image

You can use an entire image to learn the color spectrum that represents the entire color distribution of the image. In a fabric identification application, for example, an entire image can specify the color information associated with a certain fabric type, as shown in Figure 3-6.



**Figure 3-6.** Using the Entire Image to Learn Color Distribution

## Using a Region in the Image

You can select a region in the image to provide the color information for comparison. A region is helpful for pulling out the useful color information in an image. Figure 3-7 shows an example of using a region that contains the color information that is important for the application.



**Figure 3-7.** Using a Single Region to Learn Color Distribution

## Using Multiple Regions in the Image

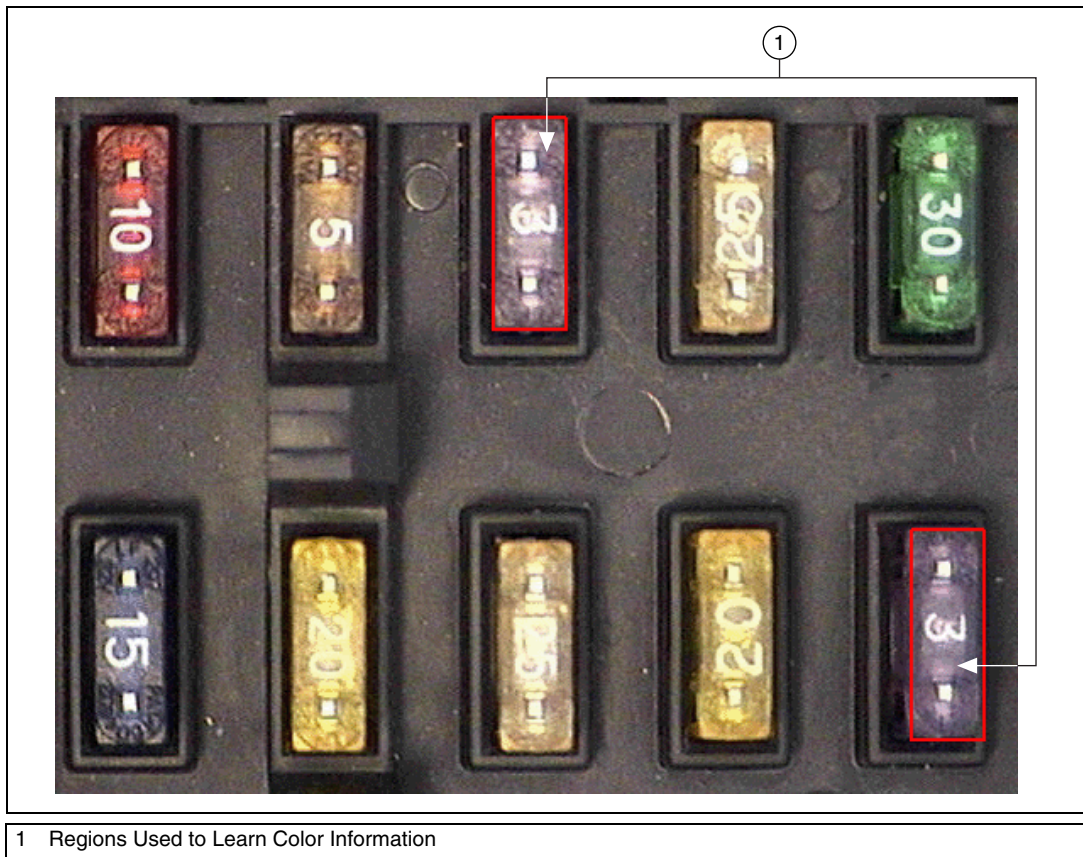
The interaction of light with the object surface creates the observed color of that object. The color of a surface depends on the directions of illumination and the direction from which the surface is observed. Two identical objects may have different appearances because of a difference in positioning or a change in the lighting conditions.

Figure 3-8 shows how light reflects differently off of the 3D surfaces of the fuses, resulting in slightly different colors for identical fuses. To view the color differences, compare the 3-amp fuse in the upper row with the 3-amp fuse in the lower row.

If you learn the color spectrum by drawing a region of interest around the 3-amp fuse in the upper row, and then do a color matching for the 3-amp fuse in the upper row, you get a very high match score for it—close to 1000. The match score for the 3-amp fuse in the lower row is low—around 500. This problem could cause a mismatch for the color matching in a fuse box inspection process.

The color learning functionality of NI Vision uses a clustering process to find the representative colors from the color information specified by one or multiple regions in the image. To create a representative color spectrum for all 3-amp fuses in the learning phase, draw a Region around the 3-amp fuse in the upper row, hold down <Ctrl>, and draw another Region around the 3 amp fuse in the lower row. The new color spectrum represents 3-amp

fuses much better and results in high match scores—around 800—for both the fuses. You can use an unlimited number of samples to learn the representative color spectrum for a specified template.



**Figure 3-8.** Using Multiple Regions to Learn Color Distribution

## Choosing a Color Representation Sensitivity

When you learn a color, you need to specify the granularity required to specify the color information. An image that contains a few well-separated colors in the color space requires a lower granularity to describe the color than an image that contains colors that are close to one another in the color space. Use the `ColorSensitivity` parameter of `CWIMAQVision.LearnColor` to specify the granularity you want to use to represent the colors. For more information about color sensitivity, refer to the [Color Sensitivity](#) section of Chapter 5, *Performing Machine Vision Tasks*.



## Ignoring Learned Colors

You can ignore certain color components in color matching by setting the corresponding component in the input color spectrum array to  $-1$ . To set a particular color component, follow these steps:

1. Copy `CWIMAQColorInformation.ColorSpectrum`, or create your own array.
2. Set the corresponding components of the array.
3. Assign this array to `CWIMAQColorInformation.ColorSpectrum` on the `CWIMAQColorInformation` object you want to use as input during the match phase.

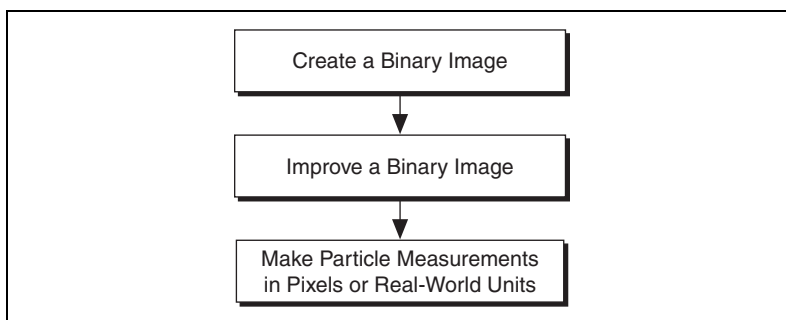
For example, setting the last component in the color spectrum to  $-1$  ignores the color white. Setting the second to last component in the color spectrum array to  $-1$  ignores the color black. To ignore other color components in color matching, determine the index to the color spectrum by locating the corresponding bins in the color wheel, where each bin corresponds to a component in the color spectrum array. Ignoring certain colors such as the background color results in a more accurate color matching score. Ignoring the background color also provides more flexibility when defining the regions of interest in the color matching process. Ignoring certain colors, such as the white color created by glare on a metallic surface, also improves the accuracy of the color matching. Experiment learning the color information about different parts of the images to determine which colors to ignore. For more information about the color wheel and color bins, refer to Chapter 15, *Color Inspection*, of the *NI Vision Concepts Manual*.

---

# Performing Particle Analysis

This chapter describes how to perform particle analysis on the images. Use particle analysis to find statistical information about particles, such as the presence, size, number, and location of particle regions. With this information, you can perform many machine vision inspection tasks, such as detecting flaws on silicon wafers or detecting soldering defects on electronic boards. Examples of how particle analysis can help you perform web inspection tasks include locating structural defects on wood planks or detecting cracks on plastic sheets.

Figure 4-1 illustrates the steps involved in performing particle analysis.



**Figure 4-1.** Steps for Performing Particle Analysis

---

## Create a Binary Image

Threshold the grayscale or color image to create a binary image. Creating a binary image separates the objects that you want to inspect from the background. The threshold operation sets the background pixels to 0 in the binary image, while setting the object pixels to a non-zero value. Object pixels have a value of 1 by default, but you can set the object pixels to any value or retain their original value.

You can use different techniques to threshold the image. If all the objects of interest in the grayscale image fall within a continuous range of intensities and you can specify this threshold range manually, use `CWIMAQVision.Threshold` to threshold the image.

If all the objects in the grayscale image are either brighter or darker than the background, you can use `CWIMAQVision.AutoThreshold2` to automatically determine the optimal threshold range and threshold the image. Automatic thresholding techniques offer more flexibility than simple thresholds based on fixed ranges. Because automatic thresholding techniques determine the threshold level according to the image histogram, the operation is more independent of changes in the overall brightness and contrast of the image than a fixed threshold. These techniques are more resistant to changes in lighting, which makes them well suited for automated inspection tasks.

If the grayscale image contains objects that have multiple discontinuous grayscale values, use `CWIMAQVision.MultiThreshold2` to specify multiple threshold ranges.

If you need to threshold a grayscale image that has nonuniform lighting conditions, such as those resulting from a strong illumination gradient or shadows, use `CWIMAQVision.LocalThreshold`. You need to define a pixel window that specifies which neighboring pixels are considered in the statistical calculation. The default window size is  $32 \times 32$ . However, the window size should be approximately the size of the smallest object you want to separate from the background. You also need to specify the local thresholding algorithm to use. The local thresholding algorithm options are the Niblack or background correction algorithm. Refer to the *NI Vision Concepts Manual* for more information about local thresholding.

If you need to threshold a color image, use `CWIMAQVision.ColorThreshold`. You must specify threshold ranges for each of the color planes—Red, Green, and Blue; or Hue, Saturation, and Luminance. The binary image resulting from a color threshold is an 8-bit binary image.

## Improve the Binary Image

---

After you threshold the image, you may want to improve the resulting binary image with binary morphology. You can use primary binary morphology or advanced binary morphology to remove unwanted particles, separate connected particles, or improve the shape of particles. Primary morphology methods work on the image as a whole by processing pixels individually. Advanced morphology operations are built upon the primary morphological operators and work on particles as opposed to pixels.

The advanced morphology methods that improve binary images require that you specify the type of *connectivity* to use. Connectivity specifies how NI Vision determines if two adjacent pixels belong to the same particle. If you have a particle that contains narrow areas, use *connectivity-8* to ensure that the software recognizes the connected pixels as one particle. If you have two particles that touch at one point, use *connectivity-4* to ensure that the software recognizes the pixels as two separate particles. For more information about connectivity, refer to Chapter 9, *Binary Morphology*, of the *NI Vision Concepts Manual*.



**Note** Use the same type of connectivity throughout the application.

## Removing Unwanted Particles

Use `CWIMAQVision.RejectBorder` to remove particles that touch the border of the image. Reject particles on the border of the image when you suspect that the information about those particles is incomplete.

Use `CWIMAQVision.RemoveParticle` to remove large or small particles that do not interest you. You also can use the `Erode`, `Open`, and `POpen` methods in `CWIMAQVision.Morphology` to remove small particles. Unlike `CWIMAQVision.RemoveParticle`, these three methods alter the size and shape of the remaining particles.

Use the hit-miss method of `CWIMAQVision.Morphology` to locate particular configurations of pixels, which you define with a structuring element. Depending on the configuration of the structuring element, the hit-miss method can locate single isolated pixels, cross-shape or longitudinal patterns, right angles along the edges of particles, and other user-specified shapes. For more information about structuring elements, refer to Chapter 9, *Binary Morphology*, of the *NI Vision Concepts Manual*.

If you know enough about the shape features of the particles you want to keep, use `CWIMAQVision.ParticleFilter3` to filter out particles that do not interest you. If you do not have enough information about the particles you want to keep at this point in the processing, use the particle measurement methods to obtain this information before applying a particle filter. Refer to the [Make Particle Measurements](#) section for more information about the measurement methods.

## Separating Touching Particles

Use watershed transform or binary morphology to separate touching particles in images.

### Using Watershed Transform

Use `CWIMAQVision.Danielsson` to transform the binary image into a grayscale distance map in which each particle pixel is assigned a gray-level value equal to its shortest Euclidean distance from the particle border. Apply `CWIMAQVision.WatershedTransform` to the distance map to find the watershed separation lines. Use `CWIMAQVision.Mask` to superimpose the watershed lines on the original image.

Refer to the *NI Vision Concepts Manual* for more information about segmenting images using a watershed transform.

### Using Binary Morphology

Use `CWIMAQVision.Separation` or apply an erosion or an open operation with `CWIMAQVision.Morphology` to separate touching objects. `CWIMAQVision.Separation` is an advanced operation that separates particles without modifying their shapes. However, erosion and open operations alter the shape of all the particles.



**Note** A separation is a time-intensive operation compared to watershed transform, erosion, or an open operation. Consider using one of the methods other than separation if speed is an issue in your application.

## Improving Particle Shapes

Use `CWIMAQVision.FillHole` to fill holes in the particles. Use `CWIMAQVision.Morphology` to perform a variety of operations on the particles. You can use the Open, Close, Proper Open, Proper Close, and auto-median operations to smooth the boundaries of the particles. Open and Proper Open smooth the boundaries of the particle by removing small isthmuses, while close widens the isthmuses. Close and Proper Close fill small holes in the particle. Auto-median removes isthmuses and fills holes. For more information about these operations, refer to Chapter 9, *Binary Morphology*, of the *NI Vision Concepts Manual*.

# Make Particle Measurements

After you create a binary image and improve it, you can make particle measurements. With these measurements you can determine the location of particles and their shape features. Use the following methods to perform particle measurements:

- `CWIMAQVision.ParticleReport`—This method returns a `CWIMAQParticleReport` object, which contains, for each particle, nine of the most commonly used measurements, including the particle area, bounding rectangle, and center of mass. The bounding rectangle is returned as one measurement, but contains four measurement elements. The center of mass is returned as one measurement, but contains two elements.
- `CWIMAQVision.ParticleMeasurement`—This method takes the measurement you want to apply to all particles, and returns an array that contains the specified measurement for each particle.

Table 4-1 lists all of the measurements that `CWIMAQVision.ParticleMeasurement` returns.

**Table 4-1.** Measurement Types

Measurement	Description
<code>cwimaqMeasurementArea</code>	Area of the particle.
<code>cwimaqMeasurementAreaByImageArea</code>	Percentage of the particle Area covering the Image Area.
<code>cwimaqMeasurementAreaByParticleAndHolesArea</code>	Percentage of the particle Area in relation to its Particle & Holes' Area.
<code>cwimaqMeasurementAverageHorizSegmentLength</code>	Average length of a horizontal segment in the particle.
<code>cwimaqMeasurementAverageVertSegmentLength</code>	Average length of a vertical segment in the particle.
<code>cwimaqMeasurementBoundingRectBottom</code>	Y-coordinate of the lowest particle point.
<code>cwimaqMeasurementBoundingRectDiagonal</code>	Distance between opposite corners of the bounding rectangle.
<code>cwimaqMeasurementBoundingRectHeight</code>	Distance between the Y-coordinate of highest particle point and the Y-coordinate of the lowest particle point.

**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementBoundingRectLeft	X-coordinate of the leftmost particle point.
cwimaqMeasurementBoundingRectRight	X-coordinate of the rightmost particle point.
cwimaqMeasurementBoundingRectTop	Y-coordinate of highest particle point.
cwimaqMeasurementBoundingRectWidth	Distance between the X-coordinate of the leftmost particle point and the X-coordinate of the rightmost particle point.
cwimaqMeasurementCenterMassX	X-coordinate of the point representing the average position of the total particle mass assuming every point in the particle has a constant density.
cwimaqMeasurementCenterMassY	Y-coordinate of the point representing the average position of the total particle mass assuming every point in the particle has a constant density.
cwimaqMeasurementCompactnessFactor	Area divided by the product of Bounding Rect Width and Bounding Rect Height.
cwimaqMeasurementConvexHullArea	Area of the smallest convex polygon containing all points in the particle.
cwimaqMeasurementConvexHullPerimeter	Perimeter of the smallest convex polygon containing all points in the particle.
cwimaqMeasurementElongationFactor	Max Feret Diameter divided by Equivalent Rect Short Side (Feret).
cwimaqMeasurementEquivalentEllipseMajorAxis	Length of the major axis of the ellipse with the same perimeter and area as the particle.
cwimaqMeasurementEquivalentEllipseMinorAxis	Length of the minor axis of the ellipse with the same perimeter and area as the particle.

**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementEquivalentEllipseMinorAxisFeret	Length of the minor axis of the ellipse with the same area as the particle, and Major Axis equal in length to the Max Feret Diameter.
cwimaqMeasurementEquivalentRectDiagonal	Distance between opposite corners of the rectangle with the same perimeter and area as the particle.
cwimaqMeasurementEquivalentRectLongSide	Longest side of the rectangle with the same perimeter and area as the particle.
cwimaqMeasurementEquivalentRectShortSide	Shortest side of the rectangle with the same perimeter and area as the particle.
cwimaqMeasurementEquivalentRectShortSideFeret	Shortest side of the rectangle with the same area as the particle, and longest side equal in length to the Max Feret Diameter.
cwimaqMeasurementFirstPixelX	X-coordinate of the highest, leftmost particle pixel.
cwimaqMeasurementFirstPixelY	Y-coordinate of the highest, leftmost particle pixel.
cwimaqMeasurementHeywoodCircularityFactor	Perimeter divided by the circumference of a circle with the same area.
cwimaqMeasurementHolesArea	Sum of the areas of each hole in the particle.
cwimaqMeasurementHolesPerimeter	Sum of the perimeters of each hole in the particle.
cwimaqMeasurementHuMoment1	The first Hu moment.
cwimaqMeasurementHuMoment2	The second Hu moment.
cwimaqMeasurementHuMoment3	The third Hu moment.
cwimaqMeasurementHuMoment4	The fourth Hu moment.
cwimaqMeasurementHuMoment5	The fifth Hu moment.
cwimaqMeasurementHuMoment6	The sixth Hu moment.



**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementHuMoment7	The seventh Hu moment.
cwimaqMeasurementHydraulicRadius	The particle area divided by the particle perimeter.
cwimaqMeasurementImageArea	Area of the image.
cwimaqMeasurementMaxFeretDiameter	Distance between the start and end of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxFeretDiameterEndX	X-coordinate of the end of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxFeretDiameterEndY	Y-coordinate of the end of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxFeretDiameterOrientation	The angle of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxFeretDiameterStartX	X-coordinate of the start of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxFeretDiameterStartY	Y-coordinate of the start of the line segment connecting the two perimeter points that are the furthest apart.
cwimaqMeasurementMaxHorizSegmentLengthLeft	X-coordinate of the leftmost pixel in the longest row of contiguous pixels in the particle.
cwimaqMeasurementMaxHorizSegmentLengthRight	X-coordinate of the rightmost pixel in the longest row of contiguous pixels in the particle.
cwimaqMeasurementMaxHorizSegmentLengthRow	Y-coordinate of all of the pixels in the longest row of contiguous pixels in the particle.

**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementMomentOfInertiaXX	The moment of inertia in the X direction twice.
cwimaqMeasurementMomentOfInertiaXXX	The moment of inertia in the X direction three times.
cwimaqMeasurementMomentOfInertiaXXY	The moment of inertia in the X direction twice and the Y direction once.
cwimaqMeasurementMomentOfInertiaXY	The moment of inertia in the X and Y directions.
cwimaqMeasurementMomentOfInertiaXYY	The moment of inertia in the X direction once and the Y direction twice.
cwimaqMeasurementMomentOfInertiaYY	The moment of inertia in the Y direction twice.
cwimaqMeasurementMomentOfInertiaYYY	The moment of inertia in the Y direction three times.
cwimaqMeasurementNormMomentOfInertiaXX	The normalized moment of inertia in the X direction twice.
cwimaqMeasurementNormMomentOfInertiaXXX	The normalized moment of inertia in the X direction three times.
cwimaqMeasurementNormMomentOfInertiaXXY	The normalized moment of inertia in the X direction twice and the Y direction once.
cwimaqMeasurementNormMomentOfInertiaXY	The normalized moment of inertia in the X and Y directions.
cwimaqMeasurementNormMomentOfInertiaXYY	The normalized moment of inertia in the X direction once and the Y direction twice.
cwimaqMeasurementNormMomentOfInertiaYY	The normalized moment of inertia in the Y direction twice.
cwimaqMeasurementNormMomentOfInertiaYYY	The normalized moment of inertia in the Y direction three times.
cwimaqMeasurementNumberOfHoles	Number of holes in the particle.

**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementNumberOfHorizSegments	Number of horizontal segments in the particle.
cwimaqMeasurementNumberOfVertSegments	Number of vertical segments in the particle.
cwimaqMeasurementOrientation	The angle of the line that passes through the particle Center of Mass about which the particle has the lowest moment of inertia.
cwimaqMeasurementParticleAndHolesArea	Percentage of the particle Area in relation to its Particle & Holes' Area.
cwimaqMeasurementPerimeter	Sum of the perimeters of each hole in the particle.
cwimaqMeasurementRatioOfEquivalentEllipseAxes	Equivalent Ellipse Major Axis divided by Equivalent Ellipse Minor Axis.
cwimaqMeasurementRatioOfEquivalentRectSides	Equivalent Rect Long Side divided by Equivalent Rect Short Side.
cwimaqMeasurementSumX	The sum of all X-coordinates in the particle.
cwimaqMeasurementSumXX	The sum of all X-coordinates squared in the particle.
cwimaqMeasurementSumXXX	The sum of all X-coordinates cubed in the particle.
cwimaqMeasurementSumXXY	The sum of all X-coordinates squared times Y-coordinates in the particle.
cwimaqMeasurementSumXY	The sum of all X-coordinates times Y-coordinates in the particle.
cwimaqMeasurementSumXYY	The sum of all X-coordinates times Y-coordinates squared in the particle.
cwimaqMeasurementSumY	The sum of all Y-coordinates in the particle.
cwimaqMeasurementSumYY	The sum of all Y-coordinates squared in the particle.

**Table 4-1.** Measurement Types (Continued)

Measurement	Description
cwimaqMeasurementSumYYY	The sum of all Y-coordinates cubed in the particle.
cwimaqMeasurementTypesFactor	Factor relating area to moment of inertia.
cwimaqMeasurementWaddelDiskDiameter	Diameter of a disk with the same area as the particle.

---

# Performing Machine Vision Tasks

This chapter describes how to perform many common machine vision inspection tasks. The most common inspection tasks are detecting the presence or absence of parts in an image and measuring the dimensions of parts to see if they meet specifications.

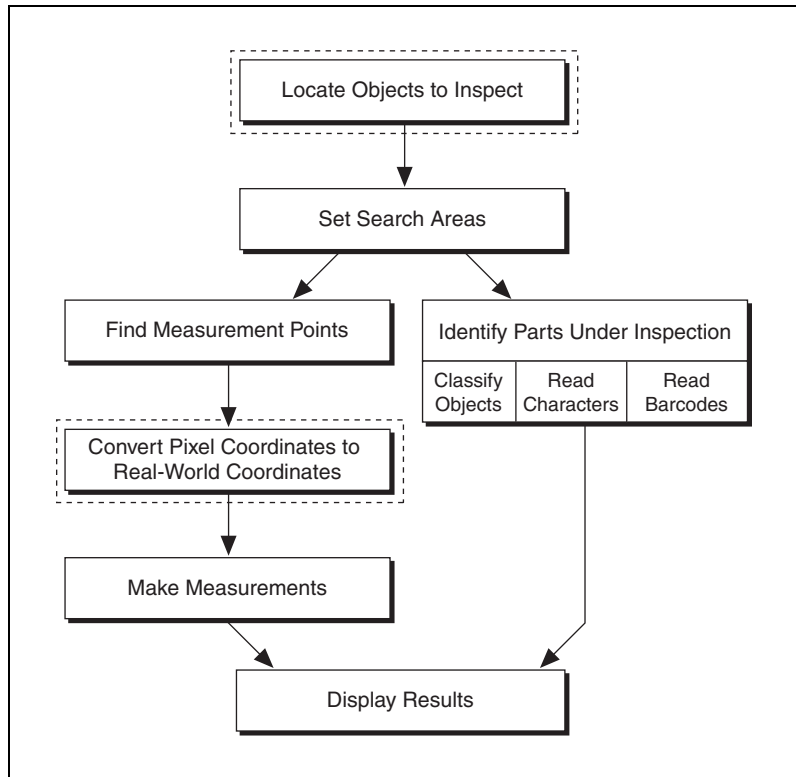
Measurements are based on characteristic features of the object represented in the image. Image processing algorithms traditionally classify the type of information contained in an image as edges, surfaces and textures, or patterns. Different types of machine vision algorithms leverage and extract one or more types of information.

Edge detectors and derivative techniques—such as rakes, concentric rakes, and spokes—use edges represented in the image. They locate, with high accuracy, the position of the edge of an object in the image. For example, you can a technique called clamping, which uses the edge location to measure the width of the part. You can combine multiple edge locations to compute intersection points, projections, circles, or ellipse fits.

Pattern matching algorithms use edges and patterns. Pattern matching can locate with very high accuracy the position of fiducials or characteristic features of the part under inspection. Those locations can then be combined to compute lengths, angles, and other object measurements.

The robustness of the measurement relies on the stability of the image acquisition conditions. Sensor resolution, lighting, optics, vibration control, part fixture, and general environment are key components of the imaging setup. All the elements of the image acquisition chain directly affect the accuracy of the measurements.

Figure 5-1 illustrates the basic steps involved in performing machine vision.



**Figure 5-1.** Steps to Performing Machine Vision



**Note** Diagram items enclosed with dashed lines are optional steps.

## Locate Objects to Inspect

In a typical machine vision application, you extract measurements from regions of interest rather than the entire image. To use this technique, the parts of the object you are interested in must always appear inside the regions of interest you define.

If the object under inspection is always at the same location and orientation in the images you need to process, defining regions of interest is simple. Refer to the [Set Search Areas](#) section of this chapter for information about selecting a region of interest.

Often, the object under inspection appears rotated or shifted in the image you need to process with respect to the reference image in which you located the object. When this occurs, the ROIs must shift and rotate with the parts of the object in which you are interested. For the ROIs to move with the object, you must define a reference coordinate system relative to the object in the reference image. During the measurement process, the coordinate system moves with the object when it appears shifted and rotated in the image you need to process. This coordinate system is referred to as the measurement coordinate system. The measurement methods automatically move the ROIs to the correct position using the position of the measurement coordinate system with respect to the reference coordinate system. For information about coordinate systems, refer to Chapter 14, *Dimensional Measurements*, of the *NI Vision Concepts Manual*.

You can build a coordinate transformation using edge detection or pattern matching. The output of the edge detection and pattern matching methods that build a coordinate transformation is a `CWMVCoordinateTransformation` object, which contains a reference coordinate system and a measurement coordinate system. Some machine vision methods take this transformation and adjust the regions of inspection automatically. You also can use these outputs to move the regions of inspection relative to the object programmatically.

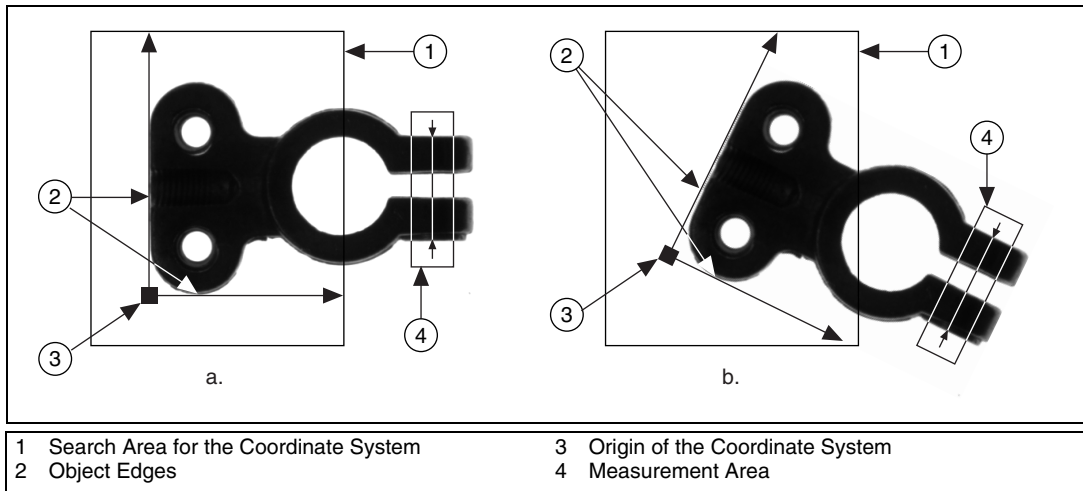
## Using Edge Detection to Build a Coordinate Transformation

You can build a coordinate transformation using two edge detection techniques. Use `CWMachineVision.FindCoordTransformUsingRect` to define a reference coordinate system using one rectangular region. Use `CWMachineVision.FindCoordTransformUsingTwoRects` to define a reference coordinate system using two independent rectangular regions. Follow the steps below to build a coordinate transformation using edge detection.



**Note** To use this technique, the object cannot rotate more than  $\pm 65^\circ$  in the image.

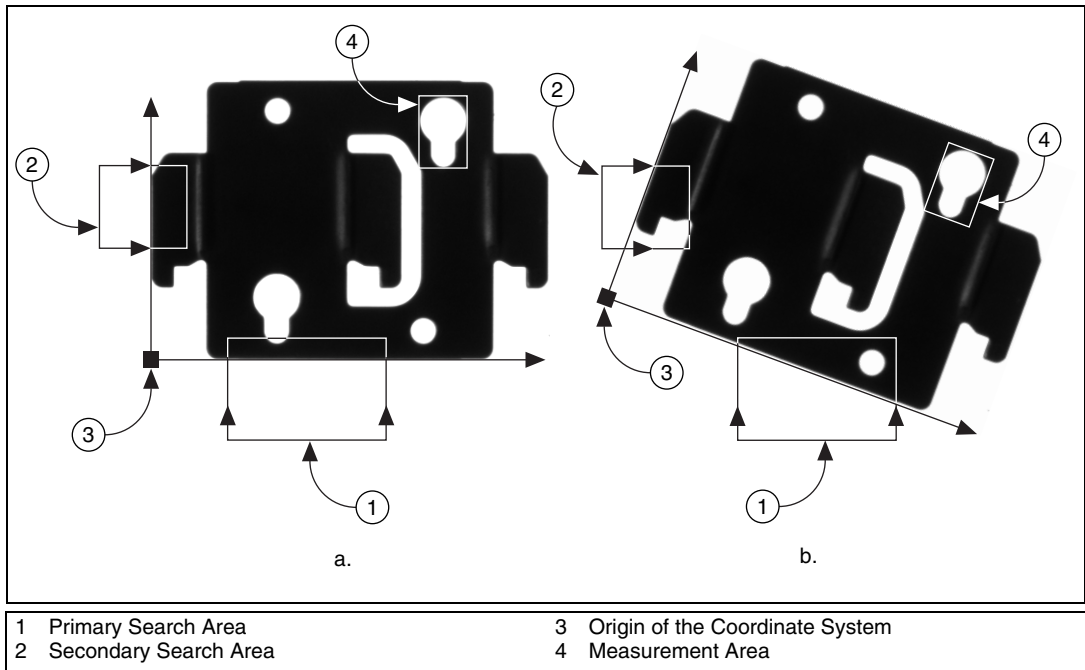
1. Specify one or two rectangular ROIs.
  - a. If you use `CWMachineVision.FindCoordTransformUsingRect`, specify one rectangular ROI that includes part of two straight, nonparallel boundaries of the object, as shown in Figure 5-2. This rectangular region must be large enough to include these boundaries in all the images you want to inspect.



**Figure 5-2.** Coordinate Systems of a Reference Image and Inspection Image

- b. If you use `CWMachineVision.FindCoordTransformUsingTwoRects`, specify two rectangular ROIs, each containing one separate, straight boundary of the object, as shown in Figure 5-3. The boundaries cannot be parallel. The regions must be large enough to include the boundaries in all of the images you want to inspect.





**Figure 5-3.** Locating Coordinate System Axes with Two Search Areas

2. Choose the parameters you need to locate the edges on the object.
3. Choose the coordinate system axis direction.
4. Choose the results that you want to overlay onto the image.
5. Choose the mode for the method. To build a coordinate transformation for the first time, set the `FirstRun` parameter to `True`. To update the coordinate transformation in subsequent images, set this parameter to `False`.

## Using Pattern Matching to Build a Coordinate Transformation

You can build a coordinate transformation using pattern matching. Use `CWMachineVision.FindCoordTransformUsingPattern` to define a reference coordinate system based on the location of a reference feature. Use this technique when the object under inspection does not have straight, distinct edges. Follow the steps below to build a coordinate transformation using pattern matching.

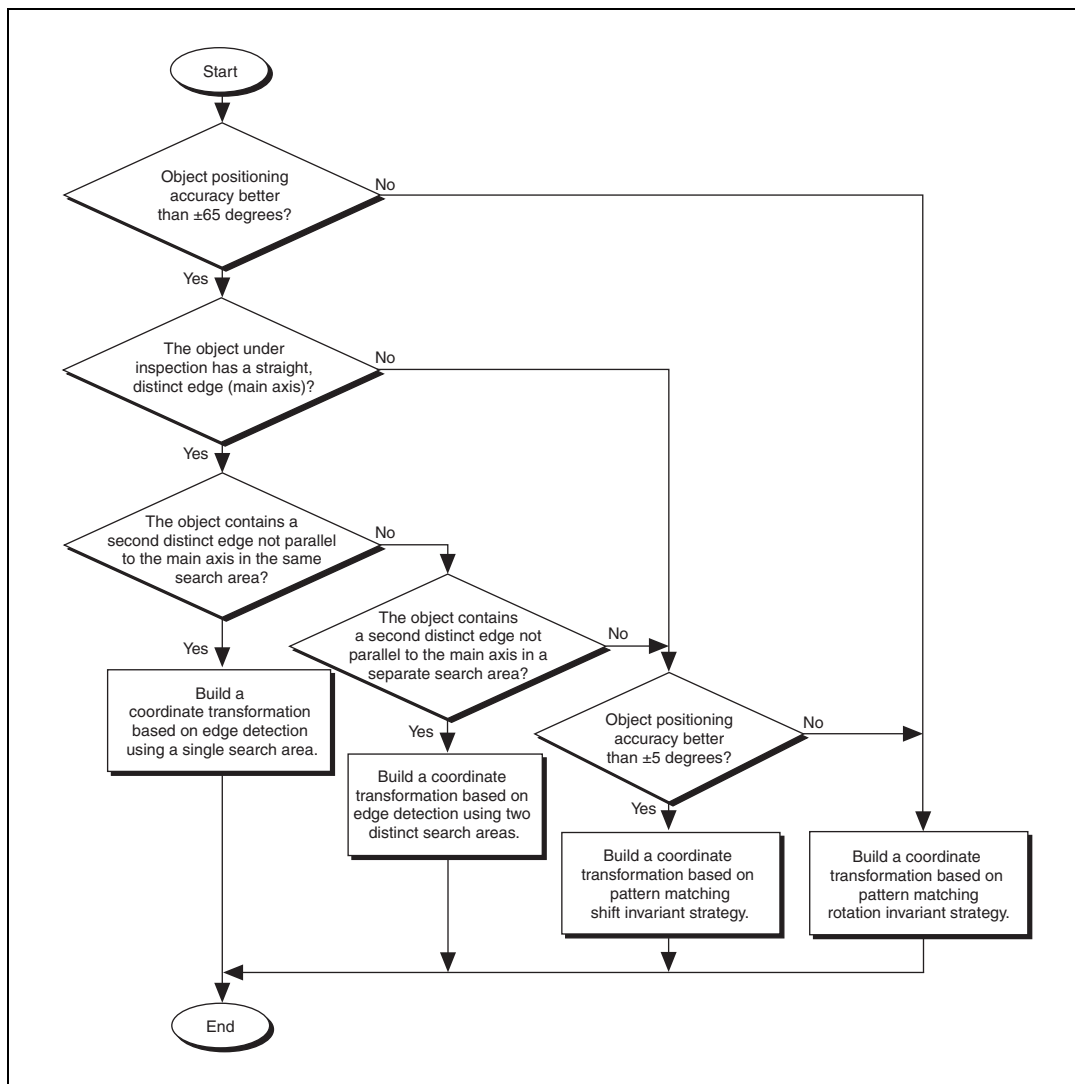


**Note** The object may rotate 360° in the image using this technique if you use rotation-invariant pattern matching.

1. Define a template that represents the part of the object that you want to use as a reference feature. For more information about defining a template, refer to the [Find Measurement Points](#) section.
2. Define a rectangular search area in which you expect to find the template.
3. Set the `MatchMode` property of the `CWMVFindCTUsingPatternOptions` object to `cwimaqRotationInvariant` when you expect the template to appear rotated in the inspection images. Otherwise, set it to `cwimaqShiftInvariant`.
4. Choose the results you want to overlay onto the image.
5. Choose the mode for the method. To build a transformation for the first time, set the `FirstRun` parameter to `True`. To update the transformation in subsequent images, set this parameter to `False`.

## Choosing a Method to Build the Coordinate Transformation

Figure 5-4 guides you through choosing the best method for building a coordinate transformation for the application.



**Figure 5-4.** Building a Coordinate Transformation

## Set Search Areas

Select ROIs in the images to limit the areas in which you perform the processing and inspection. You can define ROIs interactively or programmatically.

### Defining Regions Interactively

Follow these steps to interactively define an ROI:

1. Call `CWMachineVision.SetupViewerFor<shapename>Selection`. The following <shapename> values are available: `Annulus`, `Line`, `Point`, `Rectangle`, and `RotatedRect`. This method configures the viewer to display the appropriate tools for the shape you want to select.
2. Draw an ROI on the image. Resize or reposition the ROI until it defines the area of the image you want to process.
3. Use `CWMachineVision.GetSelected<shapename>FromViewer` to programmatically retrieve the shape from the viewer.

You also can use the techniques described in Chapter 3, *Making Grayscale and Color Measurements*, to select an ROI.

Table 5-1 indicates which ROI selection methods to use with a given `CWMachineVision` method.

**Table 5-1.** ROI Selection Methods to Use with `CWMachineVision` Methods

<b>CWMachineVision ROI Selection Methods</b>	<b>CWMachineVision Method</b>
<code>SetupViewerForRotatedRectSelection</code> <code>GetSelectedRotatedRectFromViewer</code>	<code>FindPattern</code> <code>MeasureMaximumDistance</code> <code>MeasureMinimumDistance</code> <code>FindStraightEdge</code> <code>LightMeterRectangle</code>
<code>SetupViewerForAnnulusSelection</code> <code>GetSelectedAnnulusFromViewer</code>	<code>FindCircularEdge</code> <code>FindConcentricEdge</code>
<code>SetupViewerForPointSelection</code> <code>GetSelectedPointFromViewer</code>	<code>LightMeterPoint</code>
<code>SetupViewerForLineSelection</code> <code>GetSelectedLineFromViewer</code>	<code>LightMeterLine</code>

## Defining Regions Programmatically

When you have an automated application, you need to define regions of interest programmatically. You can programmatically define regions by providing basic parameters that describe the region you want to define. You can specify a rotated rectangle by creating a `CWIMAQRotatedRectangle` object and setting the coordinates of the center, width, height, and rotation angle. You can specify an annulus by creating a `CWIMAQAnnulus` object and setting the coordinates of the center, inner radius, outer radius, start angle, and end angle. You can specify a point by setting its x-coordinates and y-coordinates. You can specify a line by setting the coordinates of the start and end points.

Refer to Chapter 3, *Making Grayscale and Color Measurements*, for more information about defining regions of interest.

## Find Measurement Points

---

After you set regions of inspection, locate points in the regions on which you can base measurements. You can locate measurement points using edge detection, pattern matching, color pattern matching, and color location.

## Finding Features Using Edge Detection

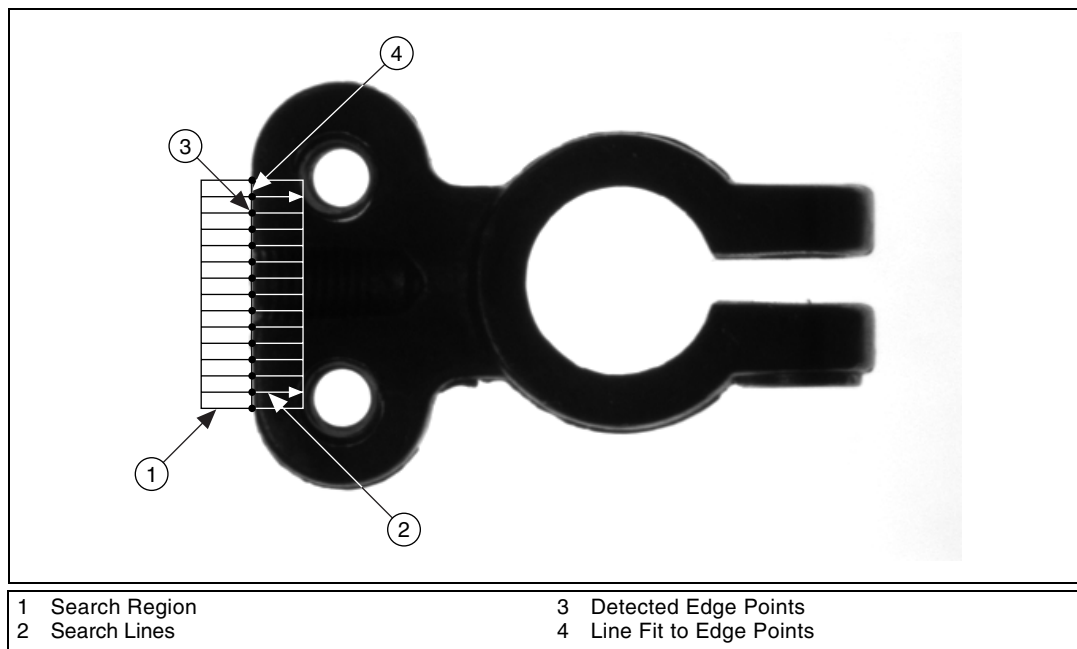
Use the edge detection tools to identify and locate sharp discontinuities in an image. Discontinuities typically represent abrupt changes in pixel intensity values, which characterize the boundaries of objects.

### Finding Lines or Circles

If you want to find points along the edge of an object and find a line describing the edge, use `CWMachineVision.FindStraightEdge` and `CWMachineVision.FindConcentricEdge`.

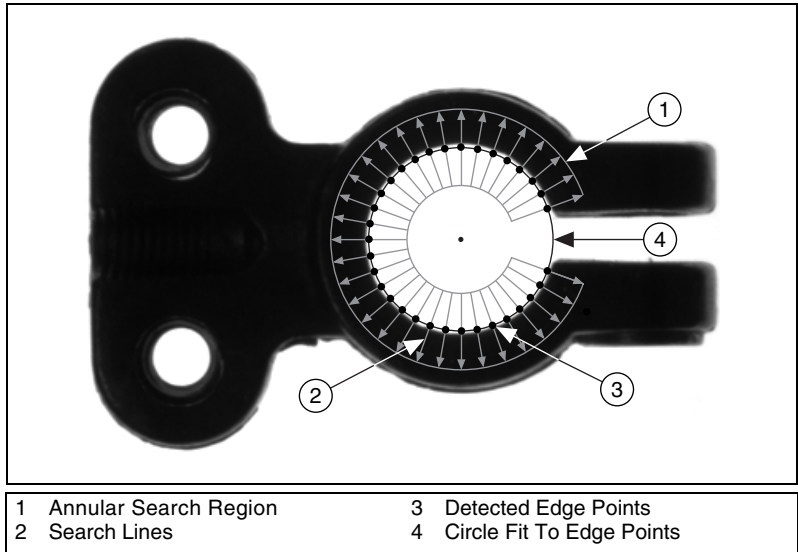
`CWMachineVision.FindStraightEdge` finds edges based on rectangular search areas, as shown in Figure 5-5.

`CWMachineVision.FindConcentricEdge` finds edges based on annular search areas.



**Figure 5-5.** Finding a Straight Feature

If you want to find points along a circular edge and find the circle that best fits the edge, as shown in Figure 5-6, use `CWMachineVision.FindCircularEdge`.



**Figure 5-6.** Finding a Circular Feature

These methods locate the intersection points between a set of search lines in the search region and the edge of an object. Specify the separation between the lines that the methods use to detect edges. The methods determine the intersection points based on their contrast, width, and steepness. The software calculates a best-fit line with outliers rejected or a best-fit circle through the points it found. The methods return the coordinates of the edges found.

## Finding Edge Points Along One Search Contour

Use `CWIMAQVision.SimpleEdge` and `CWIMAQVision.FindEdges3` to find edge points along a contour. You can find the first edge, last edge, or all edges along the contour. Use `CWIMAQVision.SimpleEdge` when the image contains little noise and the object and background are clearly differentiated. Otherwise, use `CWIMAQVision.FindEdges3`.

These methods require you to input the coordinates of the points along the search contour. Use `CWIMAQVision.RegionsProfile` to obtain the coordinates from a `CWIMAQRegions` object that describes the contour. If you have a straight line, use `CWIMAQVision.GetPointsOnLine` to obtain the points along the line instead of using regions.

These methods determine the edge points based on their contrast and slope. You can specify if you want to find the edge points using subpixel accuracy.

## Finding Edge Points Along Multiple Search Contours

Use the `CWIMAQVision.Rake`, `CWIMAQVision.Spoke`, and `CWIMAQVision.ConcentricRake` methods to find edge points along multiple search contours. These methods find only the first edge that meets the criteria along each contour. Pass in a `CWIMAQRegions` object to define the search region for these methods.

`CWIMAQVision.Rake` works on a rectangular search region. The search lines are drawn parallel to the orientation of the rectangle. Control the number of search lines in the region by specifying the distance, in pixels, between each line. Specify the search direction as left to right or right to left for a horizontally oriented rectangle. Specify the search direction as top to bottom or bottom to top for a vertically oriented rectangle.

`CWIMAQVision.Spoke` works on an annular search region, scanning the search lines that are drawn from the center of the region to the outer boundary and that fall within the search area. Control the number of lines in the region by specifying the angle, in degrees, between each line. Specify the search direction as either going from the center outward or from the outer boundary to the center.

`CWIMAQVision.ConcentricRake` works on an annular search region. The concentric rake is an adaptation of the rake to an annular region. Edge detection is performed along search lines that occur in the search region and that are concentric to the outer circular boundary. Control the number of concentric search lines that are used for the edge detection by specifying the radial distance between the concentric lines in pixels. Specify the direction of the search as either clockwise or counterclockwise.

## Finding Points Using Pattern Matching

The pattern matching algorithms in NI Vision measure the similarity between an idealized representation of a feature, called a template, and the feature that may be present in an image. A feature is defined as a specific pattern of pixels in an image. Pattern matching returns the location of the center of the template and the template orientation. Follow these generalized steps to find features in an image using pattern matching:

1. Define a reference or fiducial pattern in the form of a template image.
2. Use the reference pattern to train the pattern matching algorithm with `CWIMAQVision.LearnPattern2`.
3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.



4. Set the tolerances and parameters to specify how the algorithm operates at run time using `CWIMAQMatchPatternOptions`.
5. Test the search algorithm on test images using `CWIMAQVision.MatchPattern2`.
6. Verify the results using a ranking method.

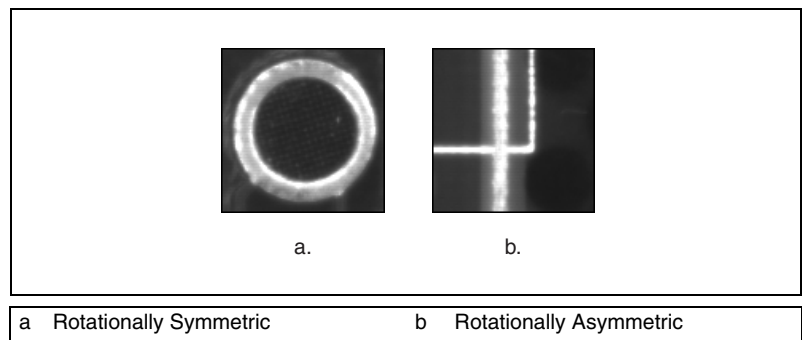
## Defining and Creating Effective Template Images

The selection of a effective template image plays a critical part in obtaining good results. Because the template image represents the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image.

Several factors are critical in creating a template image. These critical factors include symmetry, feature detail, positional information, and background information.

### Symmetry

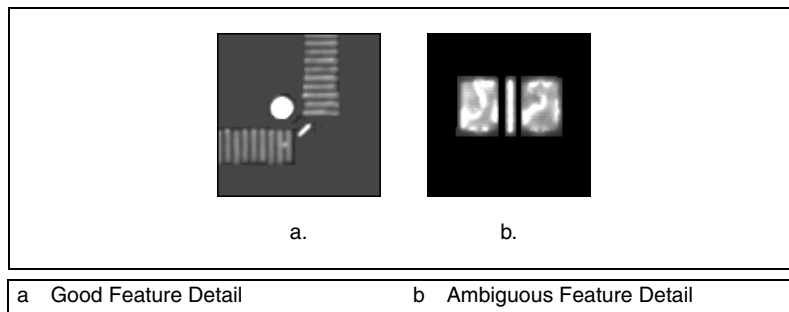
A rotationally symmetric template is less sensitive to changes in rotation than one that is rotationally asymmetric. A rotationally symmetric template provides good positioning information but no orientation information.



**Figure 5-7.** Symmetry

## Feature Detail

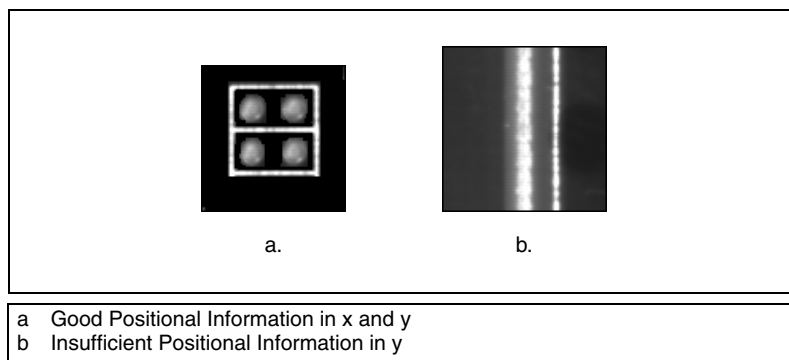
A template with relatively coarse features is less sensitive to variations in size and rotation than a model with fine features. However, the model must contain enough detail to identify it.



**Figure 5-8.** Feature Detail

## Positional Information

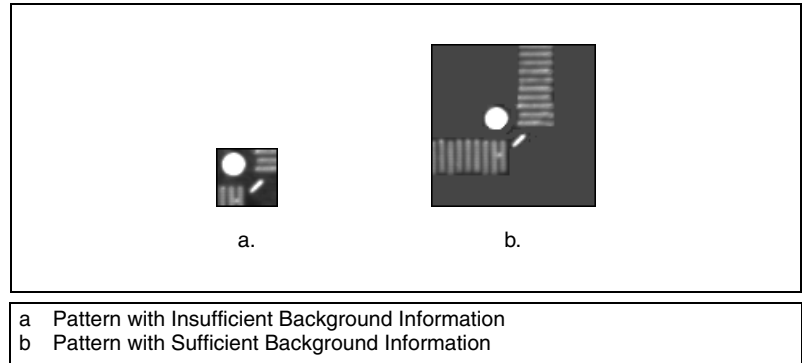
A template with strong edges in both the x and y directions is easier to locate.



**Figure 5-9.** Positional Information

## Background Information

Unique background information in a template improves search performance and accuracy.



**Figure 5-10.** Background Information

## Training the Pattern Matching Algorithm

After you create a good template image, the pattern matching algorithm has to learn the important features of the template. Use `CWIMAQVision.LearnPattern2` to learn the template. The learning process depends on the type of matching that you expect to perform. If you do not expect the instance of the template in the image to rotate or change its size, the pattern matching algorithm has to learn only those features from the template that are necessary for shift-invariant matching. However, if you want to match the template at any orientation, the learning mode must consider the possibility of arbitrary orientations. To specify which type of learning mode to use, pass the learn mode to the `LearnPatternOptions` parameter of `CWIMAQVision.LearnPattern2`. You also can set the `LearnMode` property of a `CWIMAQLearnPatternOptions` object and pass this object for the `LearnPatternOptions` parameter of `CWIMAQVision.LearnPattern2`.

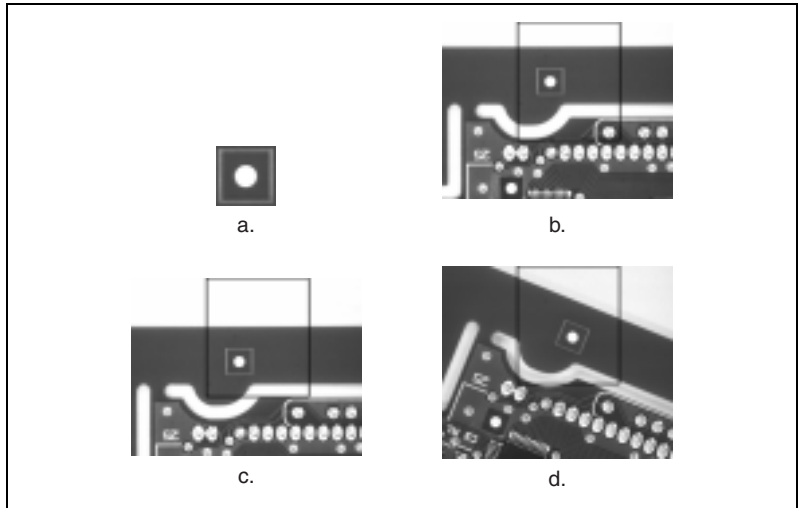
The learning process is usually time intensive because the algorithm attempts to find unique features of the template that allow for fast, accurate matching. You also can save time by training the pattern matching algorithm offline, and then saving the template image with `CWIMAQVision.WriteImageAndVisionInfo`.

## Defining a Search Area

Two equally important factors define the success of a pattern matching algorithm: accuracy and speed. You can define a search area to reduce ambiguity in the search process. For example, if the image has multiple instances of a pattern and only one of them is required for the inspection task, the presence of additional instances of the pattern can produce incorrect results. To avoid this, reduce the search area so that only the appropriate pattern lies within the search area.

The time required to locate a pattern in an image depends on both the template size and the search area. By reducing the search area or increasing the template size, you can reduce the required search time.

In many inspection applications, you have general information about the location of the fiducial. Use this information to define a search area. For example, in a typical component placement application, each printed circuit board (PCB) being tested may not be placed in the same location with the same orientation. The location of the PCB in various images can move and rotate within a known range of values, as illustrated in Figure 5-11. Figure 5-11a shows the template used to locate the PCB in the image. Figure 5-11b shows an image containing a PCB with a fiducial you want to locate. Notice the search area around the fiducial. If you know, before the matching process begins, that the PCB can shift or rotate in the image within a fixed range, as shown in Figure 5-11c and Figure 5-11d, respectively, you can limit the search for the fiducial to a small region of the image.



**Figure 5-11.** Selecting a Search Area for Grayscale Pattern Matching

## Setting Matching Parameters and Tolerances

Every pattern matching algorithm makes assumptions about the images and pattern matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications. However, there may be applications in which the assumptions used in the algorithm are not optimal. To efficiently select the best pattern matching parameters for the application, you must have a clear understanding of the application and the images you want to process. The following sections discuss parameters that influence the NI Vision pattern matching algorithm.

### Match Mode

You can set the match mode to control how the pattern matching algorithm handles the template at different orientations. If you expect the orientation of valid matches to vary less than  $\pm 5^\circ$  from the template, set `CWIMAQMatchPatternOptions.MatchMode` to `cwimaqMatchShiftInvariant`. Otherwise, set the mode element to `cwimaqMatchRotationInvariant`.



**Note** Shift-invariant matching is faster than rotation-invariant matching.

## Minimum Contrast

Contrast is the difference between the smallest and largest pixel values in a region. You can set the minimum contrast to potentially increase the speed of the pattern matching algorithm. The pattern matching algorithm ignores all image regions where contrast values fall beneath a set minimum contrast value. If the search image has high contrast but contains some low contrast regions, you can set a high minimum contrast value. Using a high minimum contrast value excludes all areas in the image with low contrast, significantly reducing the region in which the pattern matching algorithm must search. If the search image has low contrast throughout, set a low minimum contrast parameter to ensure that the pattern matching algorithm looks for the template in all regions of the image. Use `CWIMAQMatchPatternOptions.MinimumContrast` to set the minimum contrast.

## Rotation Angle Ranges

If you know that the pattern rotation is restricted to a certain range, such as between  $-15^\circ$  to  $15^\circ$ , provide this restriction information to the pattern matching algorithm in the `CWIMAQMatchPatternOptions.RotationAngleRanges` property. This information improves your search time because the pattern matching algorithm looks for the pattern at fewer angles. Refer to Chapter 12, *Pattern Matching*, of the *NI Vision Concepts Manual* for information about pattern matching.

## Testing the Search Algorithm on Test Images

To determine if the selected template or reference pattern is appropriate for the machine vision application, test the template on a few test images. These test images should reflect the images generated by the machine vision application during true operating conditions. If the pattern matching algorithm locates the reference pattern in all cases, you have selected a good template. Otherwise, refine the current template, or select a better template until both training and testing are successful.

## Using a Ranking Method to Verify Results

The manner in which you interpret the pattern matching results depends on the application. For typical alignment applications, such as finding a fiducial on a wafer, the most important information is the position and bounding rectangle of the best match. Use

`CWIMAQPatternMatchReportItem.Position` and `CWIMAQPatternMatchReportItem.BoundingPoints` to get the position and location of a match.

In inspection applications, such as optical character verification (OCV), the score of the best match is more useful. The score of a match returned by the pattern matching method is an indicator of the closeness between the original pattern and the match found in the image. A high score indicates a very close match, while a low score indicates a poor match. The score can be used as a gauge to determine if a printed character is acceptable. Use `CWIMAQPatternMatchReportItem.Score` to get a match score.

## Finding Points Using Geometric Matching

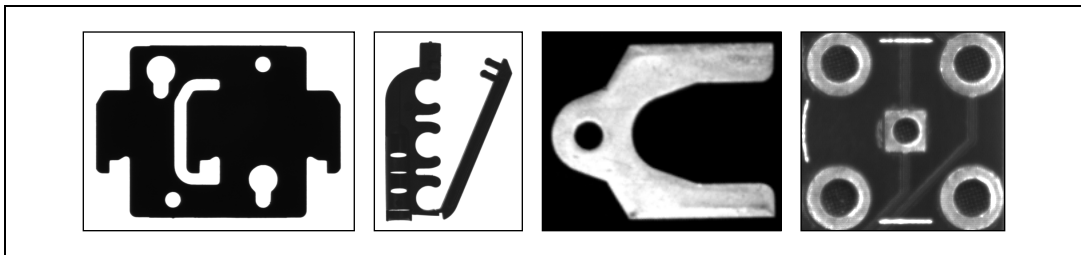
The geometric matching algorithm in NI Vision locates regions in a grayscale image that match a model, or template, of a reference pattern. Geometric matching is specialized to locate templates that are characterized by distinct geometric or shape information. Geometric matching finds template matches regardless of lighting variation, blur, noise, occlusion, and geometric transformations such as shifting, rotation, or scaling of the template. Geometric matching returns the location of the center of the template, the template orientation, and the scale of the template.

1. Define a reference or fiducial pattern to use as a template image.
2. Use the reference pattern to train the geometric matching algorithm with the NI Vision Template Editor. Go to **Start»All Programs»National Instruments»Vision»Template Editor** to launch the NI Vision Template Editor.
3. Define an image or an area of an image as the search area. A small search area can reduce the time to find the features.
4. Set the tolerances and parameters to specify how the algorithm operates at run time using `CWIMAQMatchGeometricPatternOptions`.
5. Test the search algorithm on test images using `CWIMAQVision.MatchGeometricPattern`.

## Defining and Creating Effective Template Images

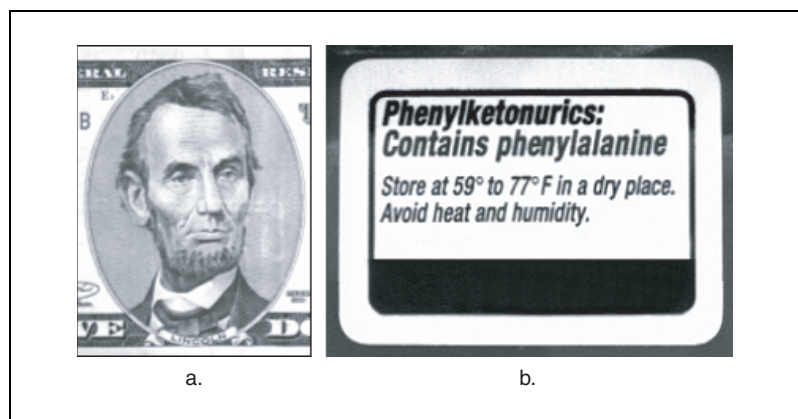
The selection of an effective template image plays a critical part in obtaining good results. Because the template image represents the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image.

The geometric matching algorithm is optimized for locating objects with good geometric information. Figure 5-12 shows examples of template images with good geometric or shape information.



**Figure 5-12.** Examples of Objects on which Geometric Matching is Designed to Work

Geometric matching is not suited for template images that are predominantly defined by grayscale or texture information. Figure 5-13 shows examples of template images that do not have good geometric information. The template image in Figure 5-13a is characterized by the grayscale or texture information in the image. Figure 5-13b contains too many edges and will dramatically increase the time geometric matching takes to locate the template in an image.



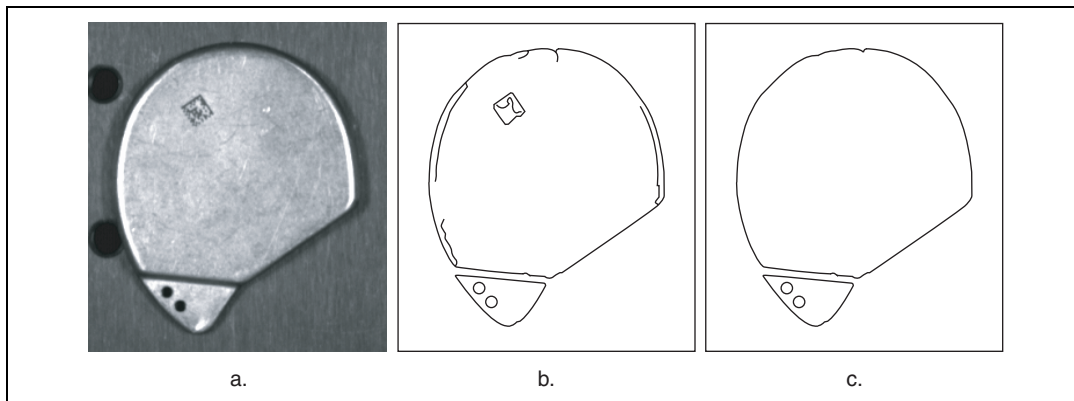
**Figure 5-13.** Examples of Objects for which Geometric Matching is Not Suited



## Training the Geometric Matching Algorithm

After you create a good template image, the geometric matching algorithm has to learn the important features of the template. Use the NI Vision Template Editor to select the important features in the template. Geometric matching uses the curves found in the template image as the basis for the features that are used for matching. Use the NI Vision Template Editor to remove noisy or unimportant curves from the template image and keep only those curves that are important to locate the template in the image.

The template image of a pacemaker is shown in Figure 5-14a. Figure 5-14b shows all the curves that are found in the template image. These images are comprised of curves that are good for matching, such as those along the boundary of the pacemaker, as well as noisy or unimportant curves, such as those found around the barcode and areas of reflection in the image. If you learn the template with the unimportant curves, the geometric matching algorithm will treat these curves as important features that must be found within a valid match region. You can make the matching process more reliable by removing these curves before learning the template. Figure 5-14c shows the curves that are used to learn the template after the unimportant curves have been removed using the NI Vision Template Editor.



**Figure 5-14.** Selecting Good Features for Training a Template

Once the template image is trained, you can save the template to a file from within the NI Vision Template Editor. Use `CWIMAQVision.ReadImageAndVisionInfo` to read the template file within your application.

Refer to the *NI Vision Template Editor Help* for more information about training the Geometric Matching algorithm.

## Setting Matching Parameters and Tolerances

The geometric matching algorithm makes assumptions about the images and geometric matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications.

However, there may be applications in which the assumptions used in the algorithm are not optimal. Knowing your particular application and the images you want to process is useful in selecting the pattern matching parameters. The following sections discuss parameters that influence the NI Vision Geometric Matching algorithm.

### Match Mode

Set the match mode to control the conditions under which the geometric matching algorithm finds the template matches. If you expect the orientation of the valid matches to vary more than  $\pm 5^\circ$  from the template, add `cwimaqGeometricMatchRotationInvariant` to `CWIMAQMatchGeometricPatternOptions.MatchMode` to enable rotation invariant matching. If you expect the size of the valid matches to change more than  $\pm 5\%$ , add `cwimaqGeometricMatchScaleInvariant` to `CWIMAQMatchGeometricPatternOptions.MatchMode` to enable scale invariant matching. If you expect the valid matches to be partially covered or missing, add `cwimaqGeometricMatchOcclusionInvariant` to `CWIMAQMatchGeometricPatternOptions.MatchMode` to enable occlusion invariant matching. Disabling any of these options or limiting their ranges decreases the search time.

### Rotation Angle Ranges

If you know the pattern is restricted to a certain rotation range—for example, between  $-15^\circ$  and  $15^\circ$ —provide this restriction information to the geometric matching algorithm in the `CWIMAQMatchGeometricPatternOptions.RotationAngleRanges` property. This information improves your search time because the geometric matching algorithm looks for the pattern at fewer angles.

### Scale Factor Ranges

When you enable scale invariant matching, geometric matching searches for occurrences of the template in the image regardless of whether valid matches are of a different size. The default scale range is 75% to 125%. If you know that the scale range of the valid matches is restricted to a certain range—for example, between 90% and 110%—provide this

restriction information to the geometric matching algorithm in the `CWIMQMatchGeometricPatternOptions.ScaleRange` property.

## Occlusion Ranges

When you enable occlusion invariant matching, geometric matching searches for occurrences of the template in the image, allowing for a specified percentage of the template to be occluded. The default occlusion range is 0% to 25%. If you know that the occlusion range of the valid matches is restricted to a certain range—for example, between 0% and 10%—provide this restriction information to the geometric matching algorithm in the

`CWIMQMatchGeometricPatternOptions.OcclusionRange` property.

Refer to the Chapter 13, *Geometric Matching*, of the *NI Vision Concepts Manual* for more information about geometric matching.

## Testing the Search Algorithm on Test Images

To determine if your selected template or reference pattern is appropriate for your machine vision application, test the template on a few test images by using `CWIMQVision.MatchGeometricPattern`. These test images should reflect the images generated by your machine vision application during true operating conditions. If the geometric matching algorithm locates the reference pattern in all cases, you have selected a good template. Otherwise, refine the current template, or select a better template until both training and testing are successful.

## Using Multiple Template Images

Complete the following steps if your application requires locating multiple template images in a target image.

1. Use the methods described in the [Defining and Creating Effective Template Images](#) and [Training the Geometric Matching Algorithm](#) sections to create and train each template image.
2. Create a `CWIMQMultipleGeometricTemplate` object. Use `CWIMQMultipleGeometricTemplate.AddTemplate` to add to this object the templates you trained in the previous step. Call `CWIMQVision.LearnMultipleGeometricPatterns` to learn the **Multiple Geometric Template**.



**Note** If you wish to perform the learning process off-line, use `CWIMAQVision.WriteMultipleGeometricTemplateFile` to save the multiple geometric templates created by this step to a file. Use `CWIMAQVision.ReadMultipleGeometricTemplateFile` to read the multiple geometric template file within your application during the matching process.

3. Use the methods described in the *Setting Matching Parameters and Tolerances* section to define how each template matches the target image. Use `CWIMAQMultipleGeometricTemplate.SetMatchOptions` to set the options for the corresponding template.



**Note** If you save the multiple geometric template to file after you set the options, these options will be loaded within your application when you load the multiple geometric template file.

4. Test the multiple template search algorithm on test images using `CWIMAQVision.MatchMultipleGeometricPatterns` to match all of the templates. The importance of this step is for the same reason explained in the *Testing the Search Algorithm on Test Images* section.

## Finding Points Using Color Pattern Matching

Color pattern matching algorithms provide a quick way to locate objects when color is present. Use color pattern matching under the following circumstances:

- The object you want to locate has color information that is very different from the background, and you want to find a very precise location of the object in the image.
- The object to locate has grayscale properties that are very difficult to characterize or that are very similar to other objects in the search image. In such cases, grayscale pattern matching can give inaccurate results. If the object has color information that differentiates it from the other objects in the scene, color provides the machine vision software with the additional information to locate the object.

Color pattern matching returns the location of the center of the template and the template orientation. Follow these general steps to find features in an image using color pattern matching:

1. Define a reference or fiducial pattern in the form of a template image.
2. Use the reference pattern to train the color pattern matching algorithm with `CWIMAQVision.LearnColorPattern`.

3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set `CWIMAQMatchColorPatternOptions.FeatureMode` to `cwimaqFeatureAll`.
5. Set the tolerances and parameters to specify how the algorithm operates at run time using `CWIMAQMatchColorPatternOptions`.
6. Test the search algorithm on test images using `CWIMAQVision.MatchColorPattern`.
7. Verify the results using a ranking method.

## Defining and Creating Effective Color Template Images

The selection of a effective template image plays a critical part in obtaining accurate results with the color pattern matching algorithm. Because the template image represents the color and the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image.

Several factors are critical in creating a template image. These critical factors include color information, symmetry, feature detail, positional information, and background information.

### Color Information

A template with colors that are unique to the pattern provides better results than a template that contains many colors, especially colors found in the background or other objects in the image.

### Symmetry

A rotationally symmetric template in the luminance plane is less sensitive to changes in rotation than one that is rotationally asymmetric.

### Feature Detail

A template with relatively coarse features is less sensitive to variations in size and rotation than a model with fine features. However, the model must contain enough detail to identify it.

### Positional Information

A template with strong edges in both the x and y directions is easier to locate.

## Background Information

Unique background information in a template improves search performance and accuracy during the grayscale pattern matching phase. This requirement could conflict with the “color information” requirement because background colors may not be appropriate during the color location phase. Avoid this problem by choosing a template with sufficient background information for grayscale pattern matching while specifying the exclusion of the background color during the color location phase. Refer to the [Training the Pattern Matching Algorithm](#) section of this chapter for more information about how to ignore colors.

## Training the Color Pattern Matching Algorithm

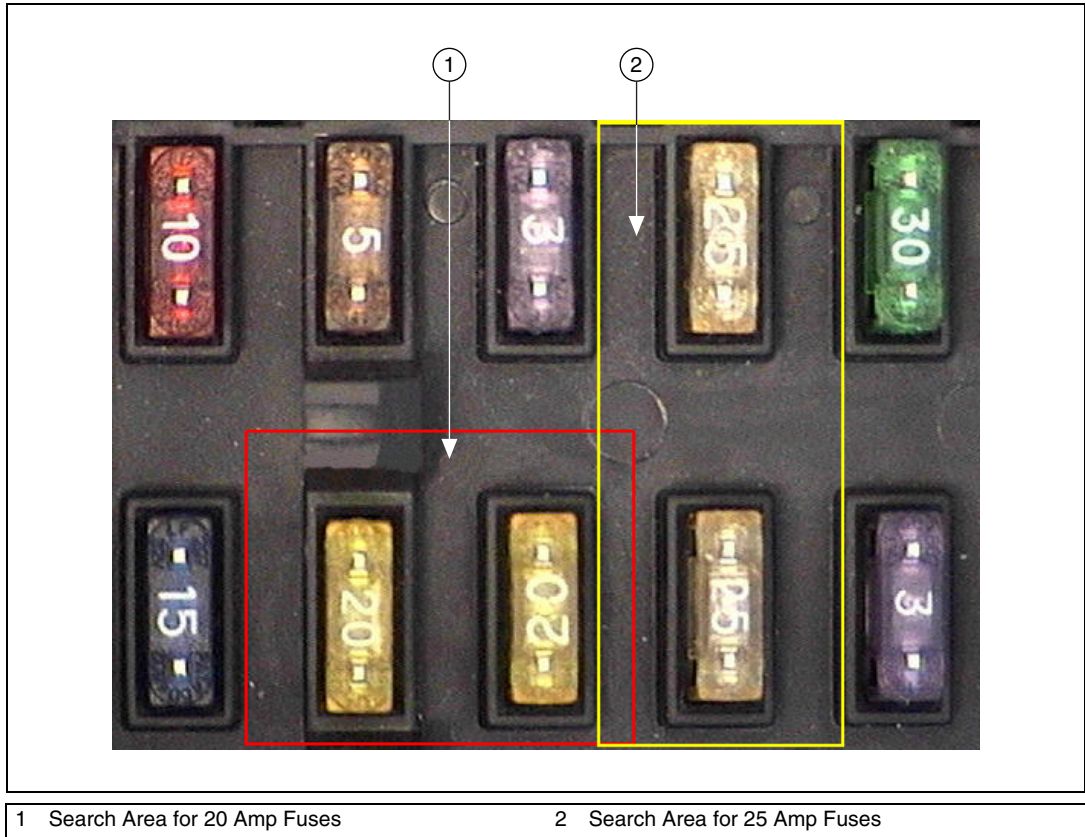
After you have created a good template image, the color pattern matching algorithm learns the important features of the template. Use `CWIMAQVision.LearnColorPattern` to learn the template. The learning process depends on the type of matching that you expect to perform. By default, the color pattern matching algorithm learns only those features from the template that are necessary for shift-invariant matching. However, if you want to match the template at any orientation, the learning process must consider the possibility of arbitrary orientations. Use the `CWIMAQLearnColorPatternOptions.LearnMode` property to specify which type of learning mode to use.

Exclude colors in the template that you are not interested in using during the search phase. Typically, you should ignore colors that either belong to the background of the object or are not unique to the template, reducing the potential for incorrect matches during the color location phase. You can learn the colors to ignore using `CWIMAQVision.LearnColor`. Use the `CWIMAQLearnColorPatternOptions.IgnoreBlackAndWhite` or `CWIMAQLearnColorPatternOptions.IgnoreColorSpectra` properties to ignore background colors.

The training or learning process is time-intensive because the algorithm attempts to find optimal features of the template for the particular matching process. However, you can train the pattern matching algorithm offline, and save the template image using `CWIMAQVision.WriteImageAndVisionInfo`.

## Defining a Search Area

Two equally important factors define the success of a color pattern matching algorithm—accuracy and speed. You can define a search area to reduce ambiguity in the search process. For example, if the image has multiple instances of a pattern and only one instance is required for the inspection task, the presence of additional instances of the pattern can produce incorrect results. To avoid this, reduce the search area so that only the appropriate pattern lies within the search area. For example, in the fuse box inspection example use the location of the fuses to be inspected to define the search area. Because the inspected fuse box may not be in the exact location or have the same orientation in the image as the previous one, the search area you define should be large enough to accommodate these variations in the position of the box. Figure 5-15 shows how search areas can be selected for different objects.



**Figure 5-15.** Selecting a Search Area for Color Pattern Matching

The time required to locate a pattern in an image depends on both the template size and the search area. By reducing the search area or increasing the template size, you can reduce the required search time. Increasing the size of the template can improve search time, but doing so reduces match accuracy if the larger template includes an excess of background information.

## Setting Matching Parameters and Tolerances

Every color pattern matching algorithm makes assumptions about the images and color pattern matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications.

In some applications, the assumptions used in the algorithm are not optimal. In such cases, you must modify the color pattern matching parameters. To efficiently select the best pattern matching parameters for the application, you must have a clear understanding of the application and the images you want to process.

The following sections discuss parameters of the NI Vision color pattern matching algorithm, and how they influence the algorithm.

### Color Sensitivity

Use the color sensitivity to control the granularity of the color information in the template image. If the background and objects in the image contain colors that are very close to colors in the template image, use a higher color sensitivity setting. A higher sensitivity setting distinguishes colors with very close hue values. Three color sensitivity settings are available in NI Vision: low, medium, and high. Use the low setting, which is the default, if the colors in the template are very different from the colors in the background or other objects that you are not interested in. Increase the color sensitivity settings as the color differences decrease. Use `CWIMAQMatchColorPatternOptions.ColorSensitivity` to set the color sensitivity. For information about color sensitivity, refer to Chapter 15, *Color Inspection*, of the *NI Vision Concepts Manual*.

### Search Strategy

Use the search strategy to optimize the speed of the color pattern matching algorithm. The search strategy controls the step size, sub-sampling factor, and the percentage of color information used from the template.



Use one of the following four search strategies:

- **Very aggressive**—Uses the largest step size, the most sub-sampling and only the dominant color from the template to search for the template. Use this strategy when the color in the template is almost uniform, the template is well contrasted from the background and there is a good amount of separation between different occurrences of the template in the image. This strategy is the fastest way to find templates in an image.
- **Aggressive**—Uses a large step size, a large amount of subsampling, and all the color spectrum information from the template.
- **Balanced**—Uses values in between the aggressive and conservative strategies.
- **Conservative**—Uses a very small step size, the least amount of subsampling, and all the color information present in the template. The conservative strategy is the most reliable method to look for a template in any image at potentially reduced speed.



**Note** Use the conservative strategy if you have multiple targets located very close to each other in the image.

Decide on the best strategy by experimenting with the different options. Use `CWIMAQMatchColorPatternOptions.SearchStrategy` to select a search strategy.

## Color Score Weight

When you search for a template using both color and shape information, the color and shape scores generated during the match process are combined to generate the final color pattern matching score. The color score weight determines the contribution of the color score to the final color pattern matching score. If the template color information is superior to its shape information, set the weight higher. For example, if you use a weight of 1000, the algorithm finds each match by using both color and shape information, and then ranks the matches based entirely on their color scores. If the weight is 0, the matches are ranked based entirely on only their shape scores. Use the `CWIMAQMatchColorPatternOptions.ColorScoreWeight` property to set the color score weight.

## Minimum Contrast

Use the minimum contrast to increase the speed of the color pattern matching algorithm. The color pattern matching algorithm ignores all image regions where grayscale contrast values fall beneath a set minimum contrast value. Use

`CWIMQMatchColorPatternMatchingOptions.MinimumContrast` to set the minimum contrast. Refer to the [Setting Matching Parameters and Tolerances](#) section of this chapter for more information about minimum contrast.

## Rotation Angle Ranges

If you know that the pattern rotation is restricted to a certain range, provide this restriction information to the pattern matching algorithm by using the `CWIMQMatchPatternOptions.RotationAngleRanges` property. This information improves the search time because the color pattern matching algorithm looks for the pattern at fewer angles. Refer to Chapter 12, *Pattern Matching*, of the *NI Vision Concepts Manual* for more information about pattern matching.

## Testing the Search Algorithm on Test Images

To determine if the selected template or reference pattern is appropriate for the machine vision application, test the template on a few test images by using the `CWIMQVision.MatchColorPattern` method. These test images should reflect the images generated by the machine vision application during true operating conditions. If the color pattern matching algorithm locates the reference pattern in all cases, you have selected a good template. Otherwise, refine the current template, or select a better template until both training and testing are successful.

## Finding Points Using Color Location

Color location algorithms provide a quick way to locate regions in an image with specific colors.

Use color location under the following circumstances:

- Requires the location and the number of regions in an image with their specific color information.
- Relies on the cumulative color information in the region, instead of the color arrangement in the region.
- Does not require the orientation of the region.

- Does not always require the location with sub-pixel accuracy.
- Does not require shape information for the region.

Complete the following steps to find features in an image using color location:

1. Define a reference pattern in the form of a template image.
2. Use the reference pattern to train the color location algorithm with `CWIMAQVision.LearnColorPattern`.
3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set `CWIMAQMatchColorPatternOptions.FeatureMode` to `cwimaqFeatureColorInformation`.
5. Set the tolerances and parameters to specify how the method operates at run time using `CWIMAQMatchColorPatternOptions`.
6. Use `CWIMAQVision.MatchColorPattern` to test the color location algorithm on test images.
7. Verify the results using a ranking method.

Use `CWIMAQVision.WriteImageAndVisionInfo` to save the template image.

## Convert Pixel Coordinates to Real-World Coordinates

---

The measurement points you located with edge detection and pattern matching are in pixel coordinates. If you need to make measurements using real-world units, use

`CWIMAQVision.ConvertPixelToRealWorldCoordinates` to convert the pixel coordinates into real-world units.

## Make Measurements

---

You can make different types of measurements either directly from the image or from points that you detect in the image.

### Distance Measurements

Use the following methods to make distance measurements for the inspection application.

Clamp methods measure the separation between two edges in a rectangular search region. First, clamp methods detect points along the two edges using

the rake method, and then they compute the distance between the points detected on the edges along each search line of the rake and return the largest or smallest distance in either the horizontal or vertical direction. The **MeasurementAxis** parameter specifies the axis along which to measure. You also need to specify the parameters for edge detection and the separation between the search lines that you want to use within the search region to find the edges. These methods work directly on the image under inspection, and they output the coordinates of all the edge points that they find. The following list describes the available clamp methods:

- `CWMachineVision.MeasureMaximumDistance`—Measures the largest separation between two edges in a rectangular search region.
- `CWMachineVision.MeasureMinimumDistance`—Measures the smallest separation between two edges in a rectangular search region.

Use `CWIMAQVision.FindPointDistances` to compute the distances between consecutive pairs of points in an array of points. You can obtain these points from the image using any one of the feature detection methods described in the [Find Measurement Points](#) section of this chapter.

## Analytic Geometry Measurements

Use the following `CWIMAQVision` methods to make geometrical measurements from the points you detect in the image:

- `FitLine`—Fits a line to a set of points and computes the equation of the line.
- `FitCircle2`—Fits a circle to a set of at least three points and computes its area, perimeter and radius.
- `FitEllipse2`—Fits an ellipse to a set of at least six points and computes its area, perimeter, and the lengths of its major and minor axis.
- `FindIntersectionPoint`—Finds the intersection point of two lines specified by their start and end points.
- `FindAngleBetweenLines`—Finds the smaller angle between two lines.
- `FindPerpendicularLine`—Finds the perpendicular line from a point to a line.
- `FindDistanceFromPointToLine`—Computes the perpendicular distance between the point and the line.
- `FindBisectingLine`—Finds the line that bisects the angle formed by two lines.

- `FindMidLine`—Finds the line that is midway between a point and a line and is parallel to the line.
- `FindPolygonArea`—Calculates the area of a polygon specified by its vertex points.

## Instrument Reader Measurements

You can make measurements based on the values obtained by meter, LCD, and barcode readers.

Use `CWIMQMeterArc.CreateFromPoints` or `CWIMQMeterArc.CreateFromLines` to calibrate a meter or gauge that you want to read. `CWIMQMeterArc.CreateFromLines` calibrates the meter using the initial position and the full-scale position of the needle. `CWIMQMeterArc.CreateFromPoints` calibrates the meter using three points on the meter: the base of the needle, the tip of the needle at its initial position, and the tip of the needle at its full-scale position. Use `CWIMQVision.ReadMeter` to read the position of the needle using the `CWIMQMeterArc` object.

Use `CWIMQVision.FindLCDSegments` to calculate the regions of interest around each digit in an LCD or LED. To find the area of each digit, all the segments of the indicator must be activated. Use `CWIMQVision.ReadLCD` to read the digits of an LCD or LED.

## Identify Parts Under Inspection

---

In addition to making measurements after you set regions of inspection, you also can identify parts using classification, OCR, and barcode reading.

## Classifying Samples

Use classification to identify an unknown object by comparing a set of its significant features to a set of features that conceptually represent classes of known objects. Typical applications involving classification include the following:

- **Sorting**—Sorts objects of varied shapes. For example, sorting different mechanical parts on a conveyor belt into different bins.
- **Inspection**—Inspects objects by assigning each object an identification score and then rejecting objects that do not closely match members of the training set.

Before you classify objects, you must create a classifier file with samples of the objects using the NI Classification Training Interface. Go to **Start»All Programs»National Instruments»Vision»Classification Training** to launch the NI Classification Training Interface.

After you have trained samples of the objects you want to classify, use the following methods to classify the image under inspection:

- Use `CWIMAQVision.ReadClassifierFile` to read in the classifier file you created with the NI Classification Training Interface.
- Use `CWIMAQClassifier.Classify` to classify the image under inspection.

## Reading Characters

Use OCR to read text and/or characters in an image. Typical uses for OCR in an inspection application include identifying or classifying components.

Before you read text and/or characters in an image, you must create a character set file with samples of the characters using the OCR Training Interface. Go to **Start»All Programs»National Instruments»Vision»OCR Training** to launch the OCR Training Interface.

After you have trained samples of the characters you want to read, use the following methods to read the characters:

- Use `NIOCR.ReadOCRFile` to read in a character set file that you created using the OCR Training Interface.
- Use `NIOCR.ReadText` to read the characters inside the ROI of the image under inspection.

## Reading Barcodes

Use barcode reading objects to read values encoded into 1D barcodes, Data Matrix codes, and PDF417 codes.

### Read 1D Barcodes

To read a 1D barcode, locate the barcode in the image using one of the techniques described in the [Instrument Reader Measurements](#) section, and then pass the `Regions` parameter into `CWIMAQVision.ReadBarcode`.

Use `CWIMAQVision.ReadBarcode` to read values encoded in the 1D barcode. Specify the type of 1D barcode in the application using the **BarcodeType** parameter. NI Vision supports the following 1D barcode

types: Codabar, Code 39, Code 93, Code 128, EAN 8, EAN 13, Interleaved 2 of 5, MSI, UPCA, RSS Limited, and Pharmacode.

## Read Data Matrix Code

Use `CWIMAQVision.ReadDataMatrixBarcode2` to read values encoded in a Data Matrix code. This method can automatically locate the Data Matrix code in an image. This method also can automatically determine the appropriate search options for the application. However, you can improve the performance of the application by specifying control values specific to the application.

## Read QR Code

Use `CWIMAQVision.ReadQRCode` to read values encoded in a QR or micro-QR code. This method can automatically locate the Data Matrix code in an image. This method also can automatically determine the appropriate search options for the application. However, you can improve the performance of the application by specifying control values specific to the application.

## Read PDF417 Code

Use `CWIMAQVision.ReadPDF417Barcode` to read values encoded in a PDF417 code. By default, `CWIMAQVision.ReadPDF417Barcode` automatically locates one or multiple PDF417 codes in an image.



**Tip** If you need to read only one code per image, set the `SearchMode` parameter to `cwimaqBarcode2DSearchSingleConservative` to increase the speed of the method.

# Inspect Image for Defects

---

You can inspect an image for defects by comparing it to a golden template and by verifying characters in the image.

## Compare to Golden Template

Use `CWIMAQVision.CompareGoldenTemplate` to inspect your image based on differences in intensity. Use the NI Vision Template Editor to learn golden templates for inspection. Go to **Start»All Programs»National Instruments»Vision»Template Editor** to launch the NI Vision Template Editor. Use the methods described in Chapter 4, *Performing Particle Analysis*, to analyze the resulting binary image.

## Verify Characters

Use character verification functions to verify characters in your image.

Before you verify text, you must train the OCR Session with samples of the characters using the NI OCR Training Interface. Go to **Start»All Programs»National Instruments»Vision»OCR Training** to launch the NI OCR Training Interface. You must then designate a reference character for each of the character classes in your character set. The characters you want to verify are compared against these reference characters and are assigned a score based on this comparison.

After you have trained the samples of the characters and assigned reference characters, use the following functions to verify the characters.

- Use `NIOCR.ReadOCRFile` to read in a character set file that you created using the OCR Training Interface.
- Use `NIOCR.VerifyText` to verify the characters inside the ROI of the image under inspection.

## Display Results

---

You can display the results obtained at various stages of the inspection process on the window that displays the inspection image by overlaying information about an image. The software attaches the information that you want to overlay to the image, but it does not modify the image.

Access overlays using the `CWIMAQImage.Overlays` property. The `CWIMAQOverlays` collection contains a single `CWIMAQOverlay` object that you can access using `CWIMAQImage.Overlay(1)`.



**Note** The `CWIMAQImage.Overlays` collection does not support usual collection methods—such as `Add`, `Remove`, and `RemoveAll`—because they are reserved for future use.

Use the following methods on the `CWIMAQOverlay` object to overlay search regions, inspection results, and other information, such as text and pictures. Overlays on a viewer image are automatically updated when you call one of these methods.

- `DrawLine`—Overlays a `CWIMAQLine` object on an image.
- `DrawConnectedPoints`—Overlays a `CWIMAQPoints` collection and draws a line between sequential points.
- `DrawRectangle`—Overlays a `CWIMAQRectangle` object on an image.



- `DrawOval`—Overlays a `CWIMAQOval` object on an image.
- `DrawArc`—Overlays a `CWIMAQArc` object on an image.
- `DrawPicture`—Overlays a picture object onto the image.
- `DrawText`—Overlays text on an image.
- `DrawRegions`—Overlays an ROI described by the `CWIMAQRegions` object on an image.



**Tip** You can select the color of overlays by using one of these methods. If you do not supply a color to an overlay method, the `CWIMAQOverlay.DefaultColor` property is used.

You can configure the following `CWMachineVision` methods to overlay different types of information about the inspection image:

- `FindStraightEdge`
- `FindCircularEdge`
- `FindConcentricEdge`
- `MeasureMaximumDistance`
- `MeasureMinimumDistance`
- `FindPattern`
- `CountAndMeasureObjects`
- `FindCoordTransformUsingRect`
- `FindCoordTransformUsingTwoRects`
- `FindCoordTransformUsingPattern`

You can overlay the following information with all the above methods except `CWMachineVision.FindPattern`:

- The search area input into the method
- The search lines used for edge detection
- The edges detected along the search lines
- The result of the method

Each of the above `CWMachineVision` methods has a settings object input that allows you to select the information you want to overlay. Set the boolean property that corresponds to the information you want to overlay to `True`. With `CWMachineVision.FindPattern`, you can overlay the search area and the result.

Use `CWIMAQOverlay.Clear` to clear any previous overlay information from the image. Use `CWIMAQVision.WriteImageAndVisionInfo` to save an image with its overlay information to a file. You can

read the information from the file into an image using the  
`CWIMAQVision.ReadImageAndVisionInfo.`



**Note** As with calibration information, overlay information is removed from an image when the image size or orientation changes.

---

# Calibrating Images

This chapter describes how to calibrate the imaging system, save calibration information, and attach calibration information to an image.

After you set up the imaging system, you may want to calibrate the system. If the imaging setup is such that the camera axis is perpendicular or nearly perpendicular to the object under inspection and the lens has no distortion, use simple calibration. With simple calibration, you do not need to learn a template. Instead, you define the distance between pixels in the horizontal and vertical directions using real-world units.

If the camera axis is not perpendicular to the object under inspection or the lens is distorted, use perspective and nonlinear distortion calibration to calibrate the system.

## Perspective and Nonlinear Distortion Calibration

---

Perspective errors and lens aberrations cause images to appear distorted. This distortion misplaces information in an image, but it does not necessarily destroy the information in the image. Calibrate the imaging system if you need to compensate for perspective errors or nonlinear lens distortion.

Follow these general steps to calibrate the imaging system:

1. Define a calibration template.
2. Define a reference coordinate system.
3. Learn the calibration information.

After you calibrate the imaging setup, you can attach the calibration information to an image. Refer to the [Attach Calibration Information](#) section of this chapter for more information. Depending on your needs, you can either apply calibration information in one of the following ways:

- Convert pixel coordinates to real-world coordinates without correcting the image
- Create a distortion-free image by correcting the image for perspective errors and lens aberrations

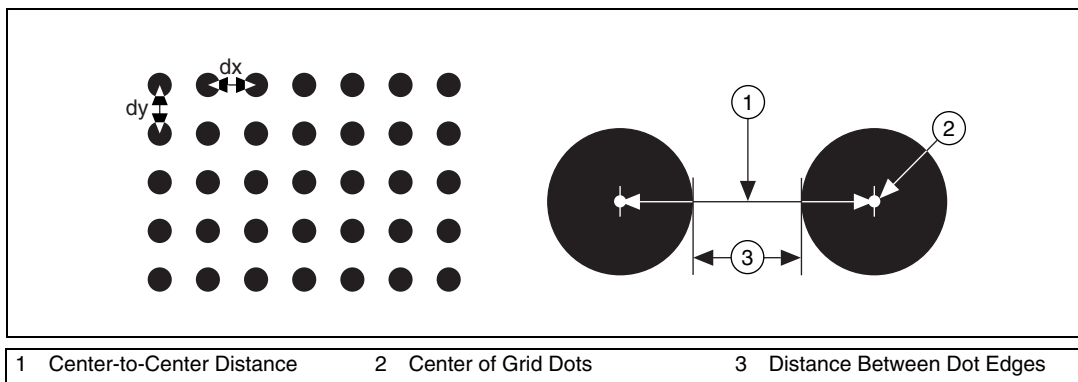
Refer to Chapter 5, *Performing Machine Vision Tasks*, for more information about applying calibration information before making measurements.

## Defining a Calibration Template

You can define a calibration template by supplying an image of a grid or providing a list of pixel coordinates and their corresponding real-world coordinates. This section discusses the grid method in detail.

A calibration template is a user-defined grid of circular dots. As shown in Figure 6-1, the grid has constant spacings in the x and y directions. You can use any grid, but follow these guidelines for the best results:

- The displacement in the x and y directions must equal ( $dx = dy$ ).
- The dots must cover the appropriate portion of the working area.
- The radius of the dots must be 6–10 pixels.
- The center-to-center distance between dots must range from 18 to 32 pixels, as shown in Figure 6-1.
- The minimum distance between the edges of the dots must be 6 pixels, as shown in Figure 6-1.



**Figure 6-1.** Defining a Calibration Grid

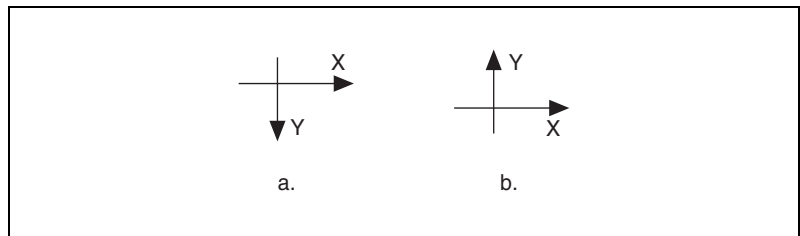


**Note** You can use the calibration grid installed with NI Vision at **Start»All Programs»National Instruments»Vision»Documentation»Calibration Grid**. The dots have radii of 2 mm and center-to-center distances of 1 cm. Depending on the printer, these measurements may change by a fraction of a millimeter. You can purchase highly accurate calibration grids from optics suppliers, such as Edmund Industrial Optics.

## Defining a Reference Coordinate System

To express measurements in real-world units, you must define a coordinate system in the image of the grid. Use `CWIMAQLearnCalibrationOptions.CalibrationAxisInfo` to define a coordinate system by its origin, angle, and axis direction.

The origin, expressed in pixels, defines the center of the coordinate system. The angle specifies the orientation of the coordinate system with respect to the angle of the topmost row of dots in the grid image. The calibration procedure automatically determines the direction of the horizontal axis in the real world. The vertical axis direction can either be indirect, as shown in Figure 6-2a, or direct, as shown in Figure 6-2b.



**Figure 6-2.** Axis Direction in the Image Plane

If you do not specify a coordinate system, the calibration process defines a default coordinate system. If you specify a grid for the calibration process, the software defines the following default coordinate system, as shown in Figure 6-3:

1. The origin is placed at the center of the left, topmost dot in the calibration grid.
2. The angle is set to  $0^\circ$ . This aligns the x-axis with the first row of dots in the grid, as shown in Figure 6-3b.
3. The axis direction is set to indirect using `CWIMAQCoordinateSystem.AxisOrientation = _cwimaqAxisOrientationIndirect`. This aligns the y-axis to the first column of the dots in the grid, as shown in Figure 6-3b.

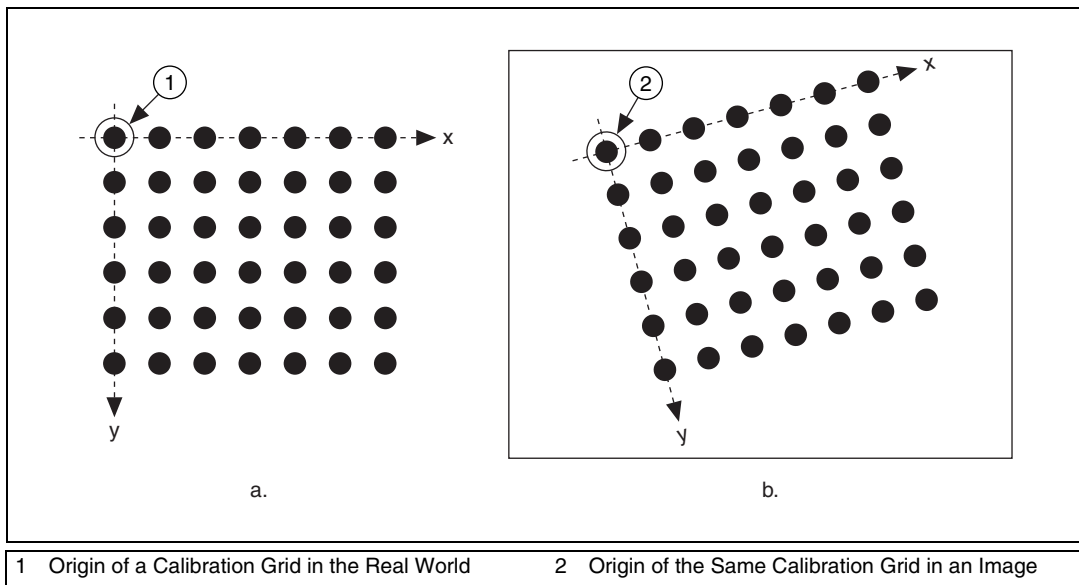


Figure 6-3. A Calibration Grid and an Image of the Grid



**Note** If you specify a list of points instead of a grid for the calibration process, the software defines a default coordinate system, as follows:

1. The origin is placed at the point in the list with the lowest x-coordinate value and then the lowest y-coordinate value.
2. The angle is set to  $0^\circ$ .
3. The axis direction is set to indirect using  
`CWIMAQCoordinateSystem.AxisOrientation = _cwimaqAxisOrientationIndirect.`

If you define a coordinate system yourself, carefully consider the requirements of the application:

- Express the origin in pixels. Always choose an origin location that lies within the calibration grid so that you can convert the location to real-world units.
- Specify the angle as the angle between the x-axis of the new coordinate system ( $x'$ ) and the top row of dots ( $x$ ), as shown in Figure 6-4. If the imaging system exhibits nonlinear distortion, you cannot visualize the angle as you can in Figure 6-4 because the dots do not appear in straight lines.

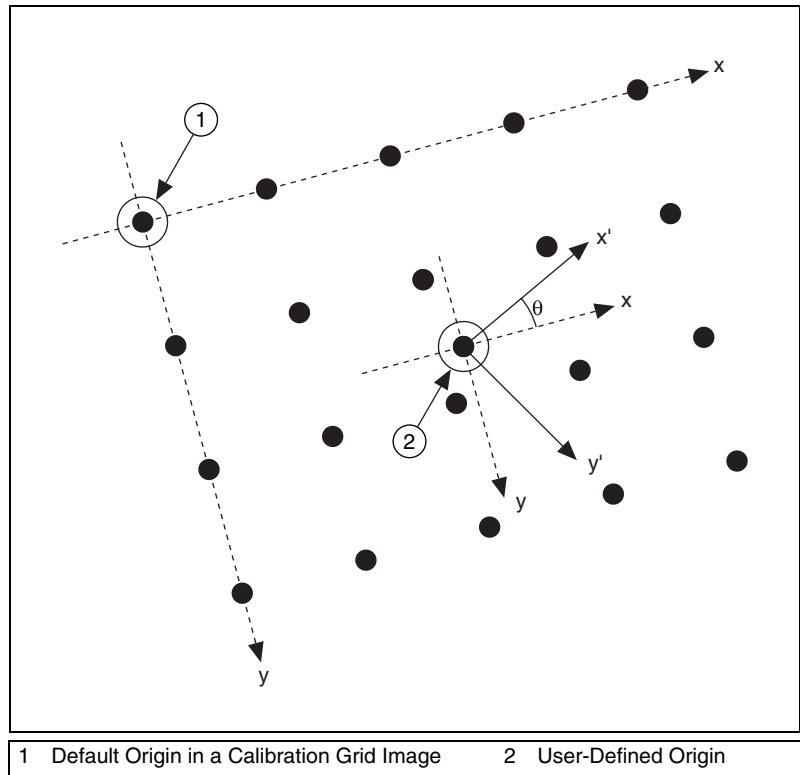


Figure 6-4. Defining a Coordinate System

## Learning Calibration Information

After you define a calibration grid and reference axis, acquire an image of the grid using the current imaging setup. For information about acquiring images, refer to the [Acquire or Read an Image](#) section of Chapter 2, [Getting Measurement-Ready Images](#). The grid does not need to occupy the entire image. You can choose a region within the image that contains the grid. After you acquire an image of the grid, learn the calibration information by inputting the image of the grid into `CWIMAVision.LearnCalibrationGrid`.



**Note** If you want to specify a list of points instead of a grid, use `CWIMAVision.LearnCalibrationPoints` to learn the calibration information. Use the `CWIMACalibrationPoints` object to specify the pixel to real-world mapping.

## Specifying Scaling Factors

Scaling factors are the real-world distances between the dots in the calibration grid in the x and y directions and the units in which the distances are measured. Use

`CWIMAQCalibrationGridOptions.GridDescriptor` to specify the scaling factors.

## Choosing a Region of Interest

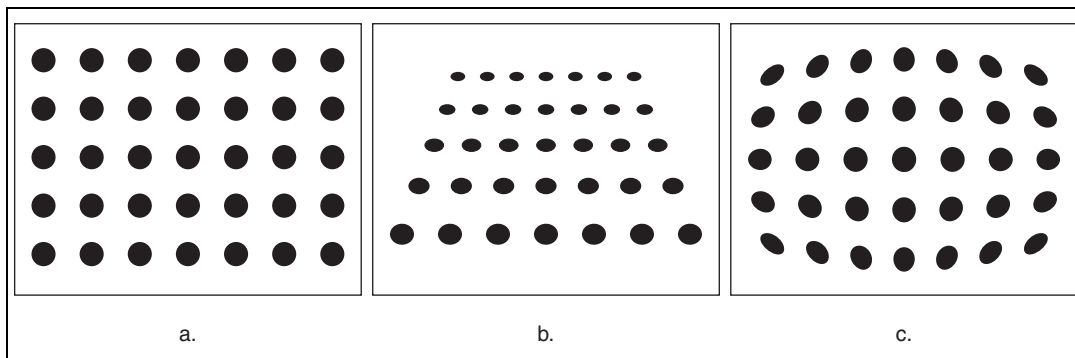
Define a learning ROI during the learning process to define a region of the calibration grid you want to learn. The software ignores dot centers outside this region when it estimates the transformation. Creating a user-defined ROI is an effective way to increase correction speeds depending on the other calibration options selected. Pass a `CWIMAQRegions` collection representing the region you want to learn to the `Regions` parameter of `CWIMAQVision.LearnCalibrationGrid` or `CWIMAQVision.LearnCalibrationPoints`.



**Note** The user-defined ROI represents the area in which you are interested. The learning ROI is separate from the calibration ROI that is generated by the calibration algorithm. Refer to Figure 6-6 for an illustration of calibration ROIs.

## Choosing a Learning Algorithm

Select a method in which to learn the calibration information: perspective projection or nonlinear. Figure 6-5 illustrates the types of errors the image can exhibit. Figure 6-5a shows an image of a calibration grid with no errors. Figure 6-5b shows an image of a calibration grid with perspective projection. Figure 6-5c shows an image of a calibration grid with nonlinear distortion.



**Figure 6-5.** Types of Image Distortion

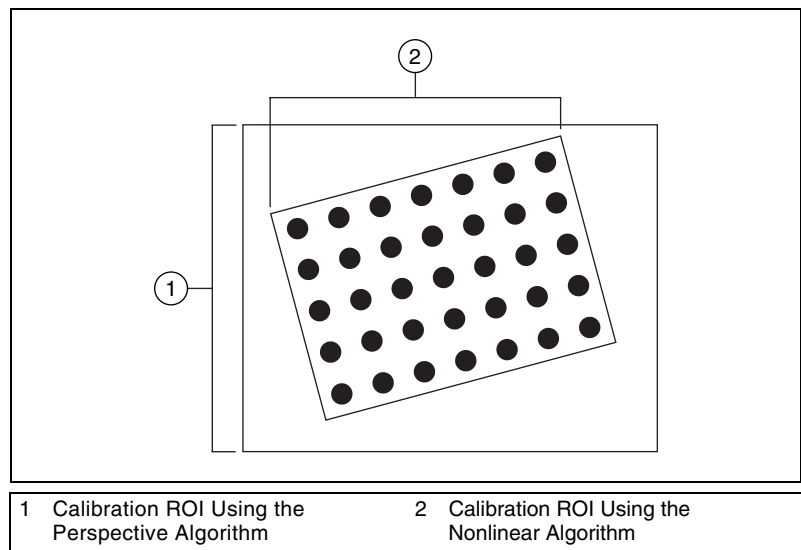


Choose the perspective projection algorithm when the system exhibits perspective errors only. A perspective projection calibration has an accurate transformation even in areas not covered by the calibration grid, as shown in Figure 6-6. Set

`CWIMAQLearnCalibrationOptions.CalibrationMethod` to `cwimaqPerspectiveCalibration` to choose the perspective calibration algorithm. Learning and applying perspective projection is less computationally intensive than the nonlinear method. However, perspective projection cannot handle nonlinear distortions.

If the imaging setup exhibits nonlinear distortion, use the nonlinear method. The nonlinear method guarantees accurate results only in the area that the calibration grid covers, as shown in Figure 6-6. If the system exhibits both perspective and nonlinear distortion, use the nonlinear method to correct for both. Set

`CWIMAQLearnCalibrationOptions.CalibrationMethod` to `cwimaqNonLinearCalibration` to choose the nonlinear calibration algorithm.



**Figure 6-6.** Calibration ROIs

## Using the Learning Score

The learning process returns a score that reflects how well the software learned the input image. A high learning score indicates that you chose the appropriate learning algorithm, that the grid image complies with the guideline, and that the vision system setup is adequate.



**Note** A high score does *not* reflect the accuracy of the system.

If the learning process returns a learning score below 600, try the following:

1. Make sure the grid complies with the guidelines listed in the [Defining a Calibration Template](#) section.
2. Check the lighting conditions. If you have too much or too little lighting, the software may estimate the center of the dots incorrectly. Also, adjust the threshold range to distinguish the dots from the background.
3. Select another learning algorithm. When nonlinear lens distortion is present, using perspective projection sometimes results in a low learning score.

## Learning the Error Map

An error map helps you gauge the quality of the complete system. The error map returns an estimated error range to expect when a pixel coordinate is transformed into a real-world coordinate. The transformation accuracy may be higher than the value the error range indicates. Set `CWIMAQLearnCalibrationOptions.LearnErrorMap` to `True` to learn the error map.

## Learning the Correction Table

If the speed of image correction is a critical factor for the application, use a correction table. The correction table is a lookup table that contains the real-world location information of all the pixels in the image. The correction table is stored in memory. The extra memory requirements for this option are based on the size of the image. Use this option when you want to simultaneously correct multiple images in the vision application. Set `CWIMAQLearnCalibrationOptions.LearnCorrectionTable` to `True` to learn the correction table.

## Setting the Scaling Mode

Use the scaling mode option to choose the appearance of the corrected image. Set

`CWIMAQLearnCalibrationOptions.CorrectionScalingMode` to `cwimaqScaleToFit` or `cwimaqScaleToPreserveArea`. For more information about the scaling mode, refer to Chapter 3, *System Setup and Calibration*, of the *NI Vision Concepts Manual*.

## Calibration Invalidation

Any image processing operation that changes the image size or orientation voids the calibration information in a calibrated image. Examples of methods that void calibration information include

`CWIMAQVision.Resample2`, `CWIMAQVision.Extract2`, `CWIMAQVision.Unwrap`, and `CWIMAQImage.ArrayToImage`.

## Simple Calibration

---

When the axis of the camera is perpendicular to the image plane and lens distortion is negligible, use simple calibration. In simple calibration, a pixel coordinate is transformed into a real-world coordinate through scaling in the horizontal and vertical directions.

Use simple calibration to map pixel coordinates to real-world coordinates directly without a calibration grid. The software rotates and scales a pixel coordinate according to predefined coordinate reference and scaling factors. You can assign the calibration to an arbitrary image using `CWIMAQVision.SetSimpleCalibration`.

To perform a simple calibration, set a coordinate system (angle, center, and axis direction) and scaling factors on the defined axis, as shown in Figure 6-7. Express the angle between the x-axis and the horizontal axis of the image in degrees. Express the center as the position, in pixels, where you want the coordinate system origin. Set the axis direction to direct or indirect. Simple calibration also offers a correction table option and a scaling mode option.

Use `CWIMAQSimpleCalibrationOptions.CalibrationAxisInfo` to define the coordinate reference. Use `CWIMAQSimpleCalibrationOptions.GridDescriptor` to specify the scaling factors. Use `CWIMAQSimpleCalibrationOptions.CorrectionScalingMode` to set the scaling mode. Set `CWIMAQSimpleCalibrationOptions.LearnCorrectionTable` to `True` to learn the correction table.

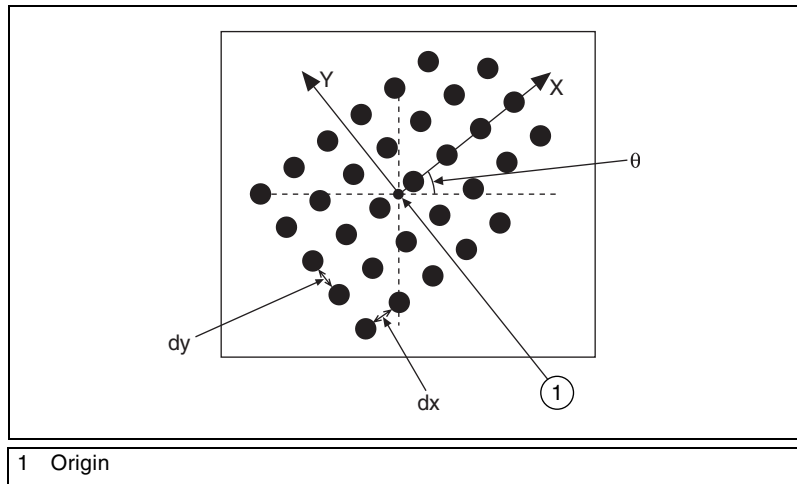


Figure 6-7. Defining a Simple Calibration

## Save Calibration Information

After you learn the calibration information, you can save it so that you do not have to relearn the information for subsequent processing. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image of the grid and its associated calibration information to a file. To read the file containing the calibration information use `CWIMAQVision.ReadImageAndVisionInfo`. For more information about attaching the calibration information you read from another image, refer to the *Attach Calibration Information* section.

## Attach Calibration Information

When you finish calibrating the setup, you can apply the calibration settings to images that you acquire. Use `CWIMAQVision.SetCalibrationInformation2` to attach the calibration information of the current setup to each image you acquire. This method takes in a source image containing the calibration information and a destination image that you want to calibrate. The output image is the inspection image with the calibration information attached to it.

Using the calibration information attached to the image, you can accurately convert pixel coordinates to real-world coordinates to make any of the analytic geometry measurements with

`CWIMAQVision.ConvertPixelToRealWorldCoordinates`. If the application requires shape measurements, correct the image by removing distortion with `CWIMAQVision.CorrectCalibratedImage`.



**Note** Correcting images is a time-intensive operation.

A calibrated image is different from a corrected image.



**Note** Because calibration information is part of the image, it is propagated throughout the processing and analysis of the image. Methods that modify the image size, such as an image rotation method, void the calibration information. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image and all of the attached calibration information to a file. If you modify the image after using `CWIMAQVision.WriteImageAndVisionInfo`, you must relearn the calibration information and use `CWIMAQVision.WriteImageAndVisionInfo` again.



---

# Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at [ni.com](http://ni.com) for technical support and professional services:

- **Support**—Online technical support resources at [ni.com/support](http://ni.com/support) include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Discussion Forums at [ni.com/forums](http://ni.com/forums). National Instruments Application Engineers make sure every question receives an answer.  
  
For information about other technical support options in your area, visit [ni.com/services](http://ni.com/services) or contact your local office at [ni.com/contact](http://ni.com/contact).
- **Training and Certification**—Visit [ni.com/training](http://ni.com/training) for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit [ni.com/alliance](http://ni.com/alliance).

If you searched [ni.com](http://ni.com) and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of [ni.com/niglobal](http://ni.com/niglobal) to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

# Glossary

---

## Numbers

1D	One-dimensional.
2D	Two-dimensional.
3D	Three-dimensional.

## A

AIPD	The National Instruments internal image file format used for saving complex images and calibration information associated with an image (extension APD).
alignment	The process by which a machine vision application determines the location, orientation, and scale of a part being inspected.
alpha channel	The channel used to code extra information, such as gamma correction, about a color image. The alpha channel is stored as the first byte in the four-byte representation of an RGB pixel.
area	(1) A rectangular portion of an acquisition window or frame that is controlled and defined by software. (2) The size of an object in pixels or user-defined units.
arithmetic operators	The image operations multiply, divide, add, subtract, and modulo.
array	An ordered, indexed set of data elements of the same type.
auto-median function	A function that uses dual combinations of opening and closing operations to smooth the boundaries of objects.

## B

b	Bit. One binary digit, either 0 or 1.
B	Byte. Eight related bits of data, an eight-bit binary number. Also denotes the amount of memory required to store one byte of data.
barycenter	The grayscale value representing the centroid of the range of an image's grayscale values in the image histogram.
binary image	An image in which the objects usually have a pixel intensity of 1 (or 255) and the background has a pixel intensity of 0.
binary morphology	Functions that perform morphological operations on a binary image.
binary threshold	The separation of an image into objects of interest (assigned a pixel value of 1) and background (assigned pixel values of 0) based on the intensities of the image pixels.
bit depth	The number of bits ( $n$ ) used to encode the value of a pixel. For a given $n$ , a pixel can take $2^n$ different values. For example, if $n$ equals 8, a pixel can take 256 different values ranging from 0 to 255. If $n$ equals 16, a pixel can take 65,536 different values ranging from 0 to 65,535 or $-32,768$ to $32,767$ .
blurring	Reduces the amount of detail in an image. Blurring commonly occurs because the camera is out of focus. You can blur an image intentionally by applying a lowpass frequency filter.
BMP	Bitmap. An image file format commonly used for 8-bit and color images. BMP images have the file extension <i>BMP</i> .
border function	Removes objects (or particles) in a binary image that touch the image border.
brightness	(1) A constant added to the red, green, and blue components of a color pixel during the color decoding process.  (2) The perception by which white objects are distinguished from gray and light objects from dark objects.
buffer	Temporary storage for acquired data.



# C

caliper	<p>(1) A function in the NI Vision Assistant and in NI Vision Builder for Automated Inspection that calculates distances, angles, circular fits, and the center of mass based on positions given by edge detection, particle analysis, centroid, and search functions.</p> <p>(2) A measurement function that finds edge pairs along a specified path in the image. This function performs an edge extraction and then finds edge pairs based on specified criteria such as the distance between the leading and trailing edges, edge contrasts, and so forth.</p>
center of mass	The point on an object where all the mass of the object could be concentrated without changing the first moment of the object about any axis.
chroma	The color information in a video signal.
chromaticity	The combination of hue and saturation. The relationship between chromaticity and brightness characterizes a color.
closing	A dilation followed by an erosion. A closing fills small holes in objects and smooths the boundaries of objects.
clustering	A technique where the image is sorted within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values and a barycenter are determined for each class. This process is repeated until a value is obtained that represents the center of mass for each phase or class.
CLUT	Color lookup table. A table for converting the value of a pixel in an image into a red, green, and blue (RGB) intensity.
color image	An image containing color information, usually encoded in the RGB form.
color space	The mathematical representation for a color. For example, color can be described in terms of red, green, and blue; hue, saturation, and luminance; or hue, saturation, and intensity.
complex image	Stores information obtained from the FFT of an image. The complex numbers that compose the FFT plane are encoded in 64-bit floating-point values: 32 bits for the real part and 32 bits for the imaginary part.

connectivity	Defines which of the surrounding pixels of a given pixel constitute its neighborhood.
connectivity-4	Only pixels adjacent in the horizontal and vertical directions are considered neighbors.
connectivity-8	All adjacent pixels are considered neighbors.
contrast	A constant multiplication factor applied to the luma and chroma components of a color pixel in the color decoding process.
convex hull	The smallest convex polygon that can encapsulate a particle.
convex hull function	Computes the convex hull of objects in a binary image.
convolution	See <a href="#">linear filter</a> .
convolution kernel	2D matrices, or templates, used to represent the filter in the filtering process. The contents of these kernels are a discrete two-dimensional representation of the impulse response of the filter that they represent.
curve extraction	The process of finding curves, or connected edge points, in a grayscale image. Curves usually represent the boundaries of objects in the image.

## D

Danielsson function	Similar to the distance functions, but with more accurate results.
determinism	A characteristic of a system that describes how consistently it can respond to external events or perform operations within a given time limit.
digital image	An image $f(x, y)$ that has been converted into a discrete number of pixels. Both spatial coordinates and brightness are specified.
dilation	Increases the size of an object along its boundary and removes tiny holes in the object.
driver	Software that controls a specific hardware device, such as an NI image acquisition or DAQ device.

## E

edge	Defined by a sharp transition in the pixel intensities in an image or along an array of pixels.
edge contrast	The difference between the average pixel intensity before and the average pixel intensity after the edge.
edge detection	Any of several techniques to identify the edges of objects in an image.
edge steepness	The number of pixels that corresponds to the slope or transition area of an edge.
energy center	The center of mass of a grayscale image. <i>See also</i> <a href="#">center of mass</a> .
equalize function	<i>See</i> <a href="#">histogram equalization</a> .
erosion	Reduces the size of an object along its boundary and eliminates isolated points in the image.
exponential and gamma corrections	Expand the high gray-level information in an image while suppressing low gray-level information.
exponential function	Decreases brightness and increases contrast in bright regions of an image, and decreases contrast in dark regions of an image.

## F

FFT	Fast Fourier Transform. A method used to compute the Fourier transform of an image.
fiducial	A reference pattern on a part that helps a machine vision application find the part's location and orientation in an image.
Fourier transform	Transforms an image from the spatial domain to the frequency domain.
frequency filters	The counterparts of spatial filters in the frequency domain. For images, frequency information is in the form of spatial frequency.
function	A set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.

## G

gamma	The nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness.
geometric features	Information extracted from a grayscale template that are used to locate the template in the target image. Geometric features in an image range from low-level features such as edges or curves detected in the image to higher-level features such as the geometric shapes made by the curves in the image.
geometric matching	A technique used to locate quickly a grayscale template that is characterized by distinct geometric or shape information within a grayscale image.
Golden Template comparison	A comparison of the pixel intensities of an image under inspection to a golden template. A golden template is an image containing an ideal representation of an object under inspection.
gradient convolution filter	<i>See</i> gradient filter.
gradient filter	An edge detection algorithm that extracts the contours in gray-level values. Gradient filters include the Prewitt and Sobel filters.
gray level	The brightness of a pixel in an image.
gray-level dilation	Increases the brightness of pixels in an image that are surrounded by other pixels with a higher intensity.
gray-level erosion	Reduces the brightness of pixels in an image that are surrounded by other pixels with a lower intensity.
grayscale image	An image with monochrome information.
grayscale morphology	Functions that perform morphological operations on a gray-level image.

# H

highpass attenuation	The inverse of lowpass attenuation.
highpass filter	Emphasizes the intensity variations in an image, detects edges or object boundaries, and enhances fine details in an image.
highpass frequency filter	Removes or attenuates low frequencies present in the frequency domain of the image. A highpass frequency filter suppresses information related to slow variations of light intensities in the spatial image.
highpass truncation	The inverse of lowpass truncation.
histogram	Indicates the quantitative distribution of the pixels of an image per gray-level value.
histogram equalization	Transforms the gray-level values of the pixels of an image to occupy the entire range of the histogram, thus increasing the contrast of the image. The histogram range in an 8-bit image is 0 to 255.
histogram inversion	Finds the photometric negative of an image. The histogram of a reversed image is equal to the original histogram flipped horizontally around the center of the histogram.
histograph	Histogram that can be wired directly into a graph.
hit-miss function	Locates objects in the image similar to the pattern defined in the structuring element.
HSI	A color encoding scheme in hue, saturation, and intensity.
HSL	A color encoding scheme using hue, saturation, and luminance information where each image in the pixel is encoded using 32 bits: 8 bits for hue, 8 bits for saturation, 8 bits for luminance, and 8 unused bits.
HSV	A color encoding scheme in hue, saturation, and value.
hue	Represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. <i>See also</i> <a href="#">RGB</a> .
Hz	Hertz. Frequency in units of 1/second.

# I

I/O	Input/output. The transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
image	A two-dimensional light intensity function $f(x, y)$ where $x$ and $y$ denote spatial coordinates and the value $f$ at any point $(x, y)$ is proportional to the brightness at that point.
image border	A user-defined region of pixels surrounding an image. Functions that process pixels based on the value of the pixel neighbors require image borders.
Image Browser	An image that contains thumbnails of images to analyze or process in a vision application.
image buffer	A memory location used to store images.
image definition	The number of values a pixel can take on, which is the number of colors or shades that you can see in the image.
image display environment	A window or control that displays an image.
image enhancement	The process of improving the quality of an image that you acquire from a sensor in terms of signal-to-noise ratio, image contrast, edge definition, and so on.
image file	A file containing pixel data and additional information about the image.
image format	Defines how an image is stored in a file. Usually composed of a header followed by the pixel data.
image mask	A binary image that isolates parts of a source image for further processing. A pixel in the source image is processed if its corresponding mask pixel has a non-zero value. A source pixel whose corresponding mask pixel has a value of 0 is left unchanged.
image palette	The gradation of colors used to display an image on screen, usually defined by a CLUT.

image processing	Encompasses various processes and analysis functions that you can apply to an image.
image source	The original input image.
imaging	Any process of acquiring and displaying images and analyzing image data.
inner gradient	Finds the inner boundary of objects.
inspection	The process by which parts are tested for simple defects such as missing parts or cracks on part surfaces.
inspection function	Analyzes groups of pixels within an image and returns information about the size, shape, position, and pixel connectivity. Typical applications include quality of parts, analyzing defects, locating objects, and sorting objects.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that control specific plug-in computer boards. Instrument drivers are available in several forms, ranging from a function callable from a programming language to a VI in LabVIEW.
intensity	The sum of the Red, Green, and Blue primary colors divided by three, $(Red + Green + Blue)/3$ .
intensity calibration	Assigns user-defined quantities such as optical densities or concentrations to the gray-level values in an image.
intensity profile	The gray-level distribution of the pixels along an ROI in an image.
intensity range	Defines the range of gray-level values in an object of an image.
intensity threshold	Characterizes an object based on the range of gray-level values in the object. If the intensity range of the object falls within the user-specified range, it is considered an object. Otherwise it is considered part of the background.

## J

jitter	The maximum amount of time that the execution of an algorithm varies from one execution to the next.
JPEG	Joint Photographic Experts Group. An image file format for storing 8-bit and color images with lossy compression. JPEG images have the file extension <i>JPG</i> .
JPEG2000	An Image file format for storing 8-bit, 16-bit, or color images with either lossy or lossless compression. JPEG2000 images have the file extension <i>JP2</i> .

## K

kernel	A structure that represents a pixel and its relationship to its neighbors. The relationship is specified by weighted coefficients of each neighbor.
--------	---

## L

labeling	A morphology operation that identifies each object in a binary image and assigns a unique pixel value to all the pixels in an object. This process is useful for identifying the number of objects in the image and giving each object a unique pixel intensity.
line gauge	Measures the distance between selected edges with high-precision subpixel accuracy along a line in an image. For example, this function can be used to measure distances between points and edges. This function also can step and repeat its measurements across the image.
line profile	Represents the gray-level distribution along a line of pixels in an image.
linear filter	A special algorithm that calculates the value of a pixel based on its own pixel value as well as the pixel values of its neighbors. The sum of this calculation is divided by the sum of the elements in the matrix to obtain a new pixel value.
local threshold	Creates a binary image by segmenting a grayscale image into a particle region and a background region.



logarithmic function	Increases the brightness and contrast in dark regions of an image and decreases the contrast in bright regions of the image.
logic operators	The image operations AND, NAND, OR, XOR, NOR, XNOR, difference, mask, mean, max, and min.
lossless compression	Compression in which the decompressed image is identical to the original image.
lossy compression	Compression in which the decompressed image is visually similar but not identical to the original image.
lowpass attenuation	Applies a linear attenuation to the frequencies in an image, with no attenuation at the lowest frequency and full attenuation at the highest frequency.
lowpass FFT filter	Removes or attenuates high frequencies present in the FFT domain of an image.
lowpass filter	Attenuates intensity variations in an image. You can use these filters to smooth an image by eliminating fine details and blurring edges.
lowpass frequency filter	Attenuates high frequencies present in the frequency domain of the image. A lowpass frequency filter suppresses information related to fast variations of light intensities in the spatial image.
lowpass truncation	Removes all frequency information above a certain frequency.
L-skeleton function	Uses an L-shaped structuring element in the skeleton function.
luma	The brightness information in the video picture. The luma signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.
luminance	<i>See</i> luma.
LUT	Lookup table. A table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table.

## M

M	<p>(1) Mega, the standard metric prefix for 1 million or <math>10^6</math>, when used with units of measure such as volts and hertz.</p> <p>(2) Mega, the prefix for 1,048,576, or 220, when used with B to quantify data or computer memory.</p>
machine vision	An automated application that performs a set of visual inspection tasks.
mask FFT filter	Removes frequencies contained in a mask (range) specified by the user.
match score	A number ranging from 0 to 1000 that indicates how closely an acquired image matches the template image. A match score of 1000 indicates a perfect match. A match score of 0 indicates no match.
MB	Megabyte of memory.
median filter	A lowpass filter that assigns to each pixel the median value of its neighbors. This filter effectively removes isolated pixels without blurring the contours of objects.
memory buffer	See <a href="#">buffer</a> .
MMX	Multimedia Extensions. An Intel chip-based technology that allows parallel operations on integers, which results in accelerated processing of 8-bit images.
morphological transformations	Extract and alter the structure of objects in an image. You can use these transformations for expanding (dilating) or reducing (eroding) objects, filling holes, closing inclusions, or smoothing borders. They are used primarily to delineate objects and prepare them for quantitative inspection analysis.
M-skeleton function	Uses an M-shaped structuring element in the skeleton function.
multiple template matching	The technique used to simultaneously locate multiple grayscale templates within a grayscale image.

## N

neighbor	A pixel whose value affects the value of a nearby pixel when an image is processed. The neighbors of a pixel are usually defined by a kernel or a structuring element.
neighborhood operations	Operations on a point in an image that take into consideration the values of the pixels neighboring that point.
NI-IMAQ	The driver software for National Instruments image acquisition hardware.
NI-IMAQdx	The National Instruments driver software for IEEE 1394 and GigE Vision cameras.
nonlinear filter	Replaces each pixel value with a nonlinear function of its surrounding pixels.
nonlinear gradient filter	A highpass edge-extraction filter that favors vertical edges.
nonlinear Prewitt filter	A highpass, edge-extraction filter based on two-dimensional gradient information.
nonlinear Sobel filter	A highpass, edge-extraction filter based on two-dimensional gradient information. The filter has a smoothing effect that reduces noise enhancements caused by gradient operators.
Nth order filter	Filters an image using a nonlinear filter. This filter orders (or classifies) the pixel values surrounding the pixel being processed. The pixel being processed is set to the $N$ th pixel value, where $N$ is the order of the filter.
number of planes (in an image)	The number of arrays of pixels that compose the image. A gray-level or pseudo-color image is composed of one plane, while an RGB image is composed of three planes (one for the red component, one for the blue, and one for the green).

## O

occlusion invariant matching	A geometric matching technique in which the reference pattern can be partially obscured in the target image.
OCR	Optical Character Recognition. The process of analyzing an image to detect and recognize characters/text in the image.

OCV	Optical Character Verification. A machine vision application that inspects the quality of printed characters.
offset	The coordinate position in an image where you want to place the origin of another image. Setting an offset is useful when performing mask operations.
opening	An erosion followed by a dilation. An opening removes small objects and smooths boundaries of objects in the image.
operators	Allow masking, combination, and comparison of images. You can use arithmetic and logic operators in NI Vision.
optical representation	Contains the low-frequency information at the center and the high-frequency information at the corners of an FFT-transformed image.
outer gradient	Finds the outer boundary of objects.

## P

palette	The gradation of colors used to display an image on screen, usually defined by a CLUT.
particle	A connected region or grouping of non-zero pixels in a binary image.
particle analysis	A series of processing operations and analysis functions that produce some information about the particles in an image.
pattern matching	The technique used to locate quickly a grayscale template within a grayscale image.
picture element	An element of a digital image. Also called pixel.
pixel	Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).
pixel aspect ratio	The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0, but typically it falls between 0.95 and 1.05, depending on camera quality.
pixel calibration	Directly calibrates the physical dimensions of a pixel in an image.

pixel depth	The number of bits used to represent the gray level of a pixel.
PNG	Portable Network Graphic. An image file format for storing 8-bit, 16-bit, and color images with lossless compression. PNG images have the file extension <i>PNG</i> .
Prewitt filter	An edge detection algorithm that extracts the contours in gray-level values using a $3 \times 3$ filter kernel.
proper-closing	A finite combination of successive closing and opening operations that you can use to fill small holes and smooth the boundaries of objects.
proper-opening	A finite combination of successive opening and closing operations that you can use to remove small particles and smooth the boundaries of objects.

## Q

quantitative analysis	Obtaining various measurements of objects in an image.
-----------------------	--

## R

real time	A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
resolution	The number of rows and columns of pixels. An image composed of m rows and n columns has a resolution of
reverse function	Inverts the pixel values in an image, producing a photometric negative of the image.
RGB	A color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 32 bits: 8 bits for red, 8 bits for green, 8 bits for blue, and 8 bits for the alpha value (unused).
RGB U64	A color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 64 bits: 16 bits for red, 16 bits for green, 16 bits for blue, and 16 bits for the alpha value (unused).
Roberts filter	An edge detection algorithm that extracts the contours in gray level, favoring diagonal edges.

ROI	Region of interest.  (1) An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing.  (2) A hardware-programmable rectangular portion of the acquisition window.
ROI tools	A collection of tools that enable you to select a region of interest from an image. These tools let you select points, lines, annuli, polygons, rectangles, rotated rectangles, ovals, and freehand open and closed contours.
rotational shift	The amount by which one image is rotated relative to a reference image. This rotation is computed relative to the center of the image.
rotation-invariant matching	A pattern matching technique in which the reference pattern can be located at any orientation in the test image as well as rotated at any degree.
<b>S</b>	
saturation	The amount of white added to a pure color. Saturation relates to the richness of a color. A saturation of zero corresponds to a pure color with no white added. Pink is a red with low saturation.
scale-invariant matching	A pattern matching technique in which the reference pattern can be any size in the test image.
segmentation function	Fully partitions a labeled binary image into non-overlapping segments, with each segment containing a unique object.
separation function	Separates objects that touch each other by narrow isthmuses.
shape detection	Detects rectangles, lines, ellipses, and circles within images.
shift-invariant matching	A pattern matching technique in which the reference pattern can be located anywhere in the test image but cannot be rotated or scaled.
skeleton function	Applies a succession of thinning operations to an object until its width becomes one pixel.
smoothing filter	Blurs an image by attenuating variations of light intensity in the neighborhood of a pixel.
Sobel filter	An edge detection algorithm that extracts the contours in gray-level values using a $3 \times 3$ filter kernel.

spatial calibration	Assigns physical dimensions to the area of a pixel in an image.
spatial filters	Alter the intensity of a pixel relative to variations in intensities of its neighboring pixels. You can use these filters for edge detection, image enhancement, noise reduction, smoothing, and so forth.
spatial resolution	The number of pixels in an image, in terms of the number of rows and columns in the image.
square function	See <a href="#">exponential function</a> .
square root function	See <a href="#">logarithmic function</a> .
standard representation	Contains the low-frequency information at the corners and high-frequency information at the center of an FFT-transformed image.
structuring element	A binary mask used in most morphological operations. A structuring element is used to determine which neighboring pixels contribute in the operation.
subpixel analysis	Finds the location of the edge coordinates in terms of fractions of a pixel.

## T

template	A color, shape, or pattern that you are trying to match in an image using the color matching, shape matching, or pattern matching functions. A template can be a region selected from an image or it can be an entire image.
threshold	Separates objects from the background by assigning all pixels with intensities within a specified range to the object and the rest of the pixels to the background. In the resulting binary image, objects are represented with a pixel intensity of 255 and the background is set to 0.
threshold interval	Two parameters, the lower threshold gray-level value and the upper threshold gray-level value.
TIFF	Tagged Image File Format. An image format commonly used for encoding 8-bit, 16-bit, and color images. TIFF images have the file extension <i>TIF</i> .
time-bounded	Describes algorithms that are designed to support a lower and upper bound on execution time.
tools palette	A collection of tools that enable you to select regions of interest, zoom in and out, and change the image palette.

## **V**

**value**                      The grayscale intensity of a color pixel computed as the average of the maximum and minimum red, green, and blue values of that pixel.

## **W**

**Watershed Transform**      Partitions an image based on the topographic surface of the image. The image is separated into non-overlapping segments with each segment containing a unique particle.



# Index

---

## Numerics

1D barcodes, reading, 5-34

## A

acquiring images, 2-4

    continuous acquisition, 2-5

    one-shot acquisition, 2-4

Acquisition Type combo box, 2-4

ActiveX objects, 1-4

adding shapes to ROIs, 3-5

analyzing images, 2-7, 2-8

Application, 1-5

application development

    general steps, 1-5

    inspection steps, 1-6

arrays, converting to images, 2-6

attaching calibration information to images, 2-7,  
    6-10

attenuation

    highpass, 2-12

    lowpass, 2-12

automatic threshold, 4-2

## B

background correction algorithm, 4-2

barcodes

    reading, 5-34

    reading 1D, 5-34

    reading data matrix barcodes, 5-35

    reading PDF417 barcodes, 5-35

binary images, improving, 4-2

binary morphology, 4-4

building

    coordinate transformation with edge  
    detection, 5-3

    coordinate transformation with pattern  
    matching, 5-5

building coordinate transformations, 5-7

    choosing a method, 5-7

## C

calibrating images, 2-2, 6-1

calibration

    nonlinear, 6-1

    perspective, 6-1

    voiding, 6-9

calibration information, 6-5

    attaching to images, 2-7, 6-10

    saving, 6-10

calibration templates, defining, 6-2

center of energy, obtaining with centroid  
    method, 3-6

centroid method, 3-6

character verification, 5-36

characters

    reading, 5-34

    training, 5-34

circle, finding points along the edge, 5-10

circles, finding, 5-9

classifying objects, 5-34

color content, evaluating in images, 3-9

color information

    learning, 3-9

    specifying, 3-10

color location, finding points, 5-30

color matching, 3-10

color pattern matching

    finding points, 5-24

    optimize speed with search strategy, 5-28

    setting rotation angle ranges, 5-30

- color pattern matching algorithms
  - training, 5-26
  - using contrast, 5-30
- color scores, 5-29
- color sensitivity, using to control granularity
  - in template images, 5-28
- color spectrums, learning, 3-10
- color statistics, measuring, 3-6, 3-7
- color template images, defining, 5-25
- color threshold, 4-2
- color, comparing in a specified region, 3-11
- colors
  - learning, 3-12
  - significant colors in the image, 3-10
- comparing colors in a specified region, 3-11
- comparison, golden template, 5-35
- complex images, 2-12
  - converting to arrays, 2-12
- continuous acquisition, 2-5
- contrast
  - color pattern matching algorithms, 5-30
  - pattern matching algorithms, 5-18
- conventions used in the manual, *ix*
- converting
  - arrays to images, 2-6
  - complex images to arrays, 2-12
  - image to its frequency domain, 2-11
  - pixel coordinates to real-world
    - coordinates, 5-31
- convolution filter, 2-10
- coordinate systems, reference, 6-3
- coordinate transformation
  - building with edge detection, 5-3
  - building with pattern matching, 5-5
- correction tables, learning, 6-8
- creating
  - binary images, 4-1, 4-2
  - images, 2-2
  - NI Vision applications, 1-4
  - template images, 5-13, 5-20

- CWIMAQ control, 1-2
- `cwimaq.ocx`, 1-1
- CWIMAQViewer control, 1-2
- CWIMAQVision, 1-2
- CWMachineVision control, 1-3
- CWMachineVision methods, 5-8
- `cwmv.ocx`, 1-3

## D

- data matrix barcodes, reading, 5-35
- defining
  - calibration templates, 6-2
  - color template images, 5-25
  - effective template images, 5-13, 5-20
  - reference coordinate systems, 6-3
  - regions interactively, 5-8
  - regions of interest, 3-1
  - regions of interest interactively, 3-1
  - regions programmatically, 5-9
  - ROIs programmatically, 3-5
  - ROIs with masks, 3-6
  - search area, 5-27
  - search areas, 5-16
  - templates with colors that are unique to
    - the pattern, 5-25
- detecting objects, 5-2
- diagnostic tools (NI resources), A-1
- displaying
  - images, 2-6
  - results, 5-36
- distance measurements, 5-31
- documentation
  - conventions used in manual, *ix*
  - NI resources, A-1
  - related documentation, *x*
- drivers (NI resources), A-1

## E

- edge detection, 5-3
  - finding features, 5-9
- edge points, finding along multiple search contours, 5-12
- error map, learning, 6-8
- evaluating color content of an image, 3-9
- examples (NI resources), A-1

## F

- features, finding with edge detection, 5-9
- FFT, 2-11
- files, reading, 2-6
- filtering
  - grayscale features, 2-10
  - images, 2-9, 2-10
- filters, FFT, 2-11
- finding
  - circles, 5-9
  - edge points along multiple search contours, 5-12
  - edge points along one search contour, 5-11
  - features with edge detection, 5-9
  - lines, 5-9
  - measurement points, 5-9
  - points along the edge of a circle, 5-10
  - points using color pattern matching, 5-24
  - points using geometric matching, 5-19
  - points using pattern matching, 5-12
  - points with color location, 5-30

## G

- geometric matching
  - finding points, 5-19
  - match mode, 5-22
  - occlusion ranges, 5-23
  - rotation angle ranges, 5-22
  - scale factor ranges, 5-22

- setting parameters and tolerances, 5-22
  - testing search algorithm, 5-23
  - training, 5-21
- geometrical measurements, 5-32
- golden template comparison, 5-35
- granularity
  - specifying requirements for learning a color, 3-12
  - using color sensitivity to control, 5-28
- grayscale features, filtering, 2-10
- grayscale morphology, filtering grayscale features, 2-10
- grayscale statistics, measuring, 3-6

## H

- help, technical support, A-1
- highpass
  - attenuation, 2-12
  - filter, 2-9

## I

- image mask, 3-6
- images
  - acquiring, 2-4
  - attaching calibration information, 2-7, 6-10
  - calibrating, 2-2
  - complex, 2-12
  - creating, 2-2
  - displaying, 2-6
  - evaluating color content, 3-9
  - filtering, 2-9, 2-10
  - filtering grayscale features, 2-10
  - highlighting details using LUTs, 2-9
  - improving, 2-9
  - measuring, 5-31
  - reading, 2-4
  - signal-to-noise ratio, 2-9
  - transitions, 2-9

- imaging systems, setting up, 2-1
- improving
  - binary images, 4-2
  - images, 2-9
  - particle shapes, 4-4
- increasing
  - speed of the color pattern matching algorithm, 5-30
  - speed of the pattern matching algorithm, 5-18
- inspection, defects, 5-35
- instrument drivers (NI resources), A-1
- instrument reader measurements, 5-33
- interactively defining regions, 5-8

## K

- KnowledgeBase, A-1

## L

- learning
  - calibration information, 6-5
  - color information, 3-9
  - color spectrums, 3-10
  - colors, 3-12
  - correction tables, 6-8
  - error maps, 6-8
- learning algorithm, specifying, 6-6
- learning calibration information
  - correction tables, 6-8
  - error maps, 6-8
  - setting the scaling mode, 6-8
  - specifying a learning algorithm, 6-6
  - specifying a region of interest, 6-6
  - specifying scaling factors, 6-6
  - using learning scores, 6-7
  - voiding calibrations, 6-9
- learning score, using, 6-7
- light intensity, measuring, 3-6
- lighting effects on image colors, 3-11

- lines, finding, 5-9
- local threshold, 4-2
- locating objects to detect, 5-2
- lowpass
  - attenuation, 2-12
  - filter, 2-9
- LUTs, highlighting details in images, 2-9

## M

- machine vision, 5-1
- masks, defining regions of interest, 3-6
- measurement points, finding, 5-9
- measurements
  - distance, 5-31
  - geometry, 5-32
  - instrument reader, 5-33
- measuring
  - color statistics, 3-6, 3-7
  - grayscale statistics, 3-6
  - images, 5-31
  - light intensity, 3-6
  - particles, 4-5
- measuring particles, 4-5
- method for building coordinate transformations, 5-7
- multiple ROIs, using to view color differences in an image, 3-11
- multiple search contours, finding edge points, 5-12
- multiple template images, 5-23
- multiple threshold, 4-2

## N

- National Instruments support and services, A-1
- NI Vision applications, creating, 1-4
- NI Vision Template Editor, 5-21
- Niblack algorithm, 4-2
- NI-IMAQ, *xi*

niocr.ocx, 1-2

nonlinear calibration, 6-1

Nth order filter, 2-10

## O

objects

- classifying, 5-34

- detecting, 5-2

- locating, 5-2

OCR, 5-34

one-shot acquisition, 2-4

optimizing speed of the color pattern matching algorithm, 5-28

## P

particle analysis, 4-1

- improving binary image

- separating touching particles, 4-4

- performing, 4-1

particle measurements, 4-5

particle shapes, improving, 4-4

particles

- measuring, 4-5

- removing unwanted, 4-3

pattern matching

- building a coordinate transformation, 5-5

- finding points, 5-12

- score, 5-29

- setting rotation angle ranges, 5-18

- setting tolerances, 5-17, 5-28

- tolerances, setting, 5-28

- training algorithm, 5-15

- verifying results, 5-19

pattern matching algorithms

- using contrast, 5-18

PDF417 barcodes, reading, 5-35

perspective calibration, 6-1

pixel coordinates, converting to real-world coordinates, 5-31

points

- finding along one search contour, 5-11

- finding along the edge of a circle, 5-10

- finding measurement points, 5-9

- finding with color location, 5-30

- finding with color pattern matching, 5-24

- finding with geometric matching, 5-19

- finding with pattern matching, 5-12

programmatically defining

- regions, 5-9

- regions of interest, 3-5

programming examples (NI resources), A-1

## R

ranking, verifying pattern matching

- results, 5-19

reading

- barcodes, 5-34

- characters, 5-34

- files, 2-6

- AIPD, 2-6

- BMP, 2-6

- JPEG, 2-6

- JPEG2000, 2-6

- PNG, 2-6

- TIFF, 2-6

- images, 2-4

reference coordinate systems, 6-3

- defining, 6-3

region of interest, specifying, 6-6

regions

- defining interactively, 5-8

- programmatically defining, 5-9

regions of interest

- defining, 3-1

- defining interactively, 3-1

- defining with masks, 3-6

related documentation, *x*

removing unwanted particles, 4-3

- results
  - displaying, 5-36
  - verifying for pattern matching, 5-19
- ROI selection methods, 5-8
- ROIs
  - adding shapes, 3-5
  - programmatically defining, 3-5
- rotation angle ranges
  - setting for color pattern matching, 5-30
  - setting for pattern matching, 5-18
- rotationally symmetric template, 5-25

## S

- saving calibration information, 6-10
- scaling factors, specifying, 6-6
- scaling mode, setting, 6-8
- search algorithms, testing on test images, 5-18, 5-30
- search area, defining, 5-16, 5-27
- search areas, setting, 5-8
- search contour, finding points, 5-11
- search contours, finding edge points along
  - multiple search contours, 5-12
- search strategy, using to optimize the speed of
  - color pattern matching algorithms, 5-28
- selecting significant colors for the object, 3-10
- separating touching particles, 4-4
- setting
  - pattern matching tolerances, 5-17, 5-28
  - rotation angle ranges for pattern matching, 5-18, 5-30
  - scaling mode, 6-8
  - search areas, 5-8
- setting up measurement systems, 2-1
- shape scores, 5-29
- signal-to-noise ratio, 2-9
- simple calibration, 6-9
- software (NI resources), A-1

- specifying
  - color information, 3-10
  - granularity to learn a color, 3-12
  - learning algorithm, 6-6
  - region of interest, 6-6
  - scaling factors, 6-6
- specifying scaling factors, learning calibration information, 6-6
- speed
  - increasing for color pattern matching algorithms, 5-30
  - increasing for pattern matching algorithms, 5-18
- support, technical, A-1
- symmetric templates, 5-13

## T

- template
  - background information, 5-15
  - coarse features, 5-14
  - defining with colors that are unique to the pattern, 5-25
  - strong edges, 5-14
- Template Editor, 5-21
- template images
  - creating, 5-13, 5-20
  - defining, 5-13, 5-20
  - using color sensitivity to control granularity, 5-28
- templates
  - background information, 5-26
  - calibration, 6-2
  - coarse features, 5-25
  - detail, 5-25
  - positional information, 5-25
  - strong edges, 5-25
- test images, testing search algorithms, 5-18, 5-30

- testing
  - geometric search algorithm, 5-23
  - search algorithms, 5-18, 5-30
- threshold
  - automatic, 4-2
  - color, 4-2
  - local, 4-2
  - multiple ranges, 4-2
- tolerances, setting for pattern matching, 5-17
- touching particles, separating, 4-4
- training
  - characters, 5-34
  - color pattern matching algorithms, 5-26
  - pattern matching algorithm, 5-15
- training and certification (NI resources), A-1
- training the geometric matching
  - algorithm, 5-21
- troubleshooting (NI resources), A-1

## U

- using
  - learning scores, 6-7
  - multiple template images, 5-23
  - ranking to verify pattern matching
    - results, 5-19

## V

- viewing color differences in an image using
  - multiple regions, 3-11
- Vision for Visual Basic organization, 1-1
- voiding calibrations, 6-9

## W

- watershed transform, 4-4
- Web resources, A-1